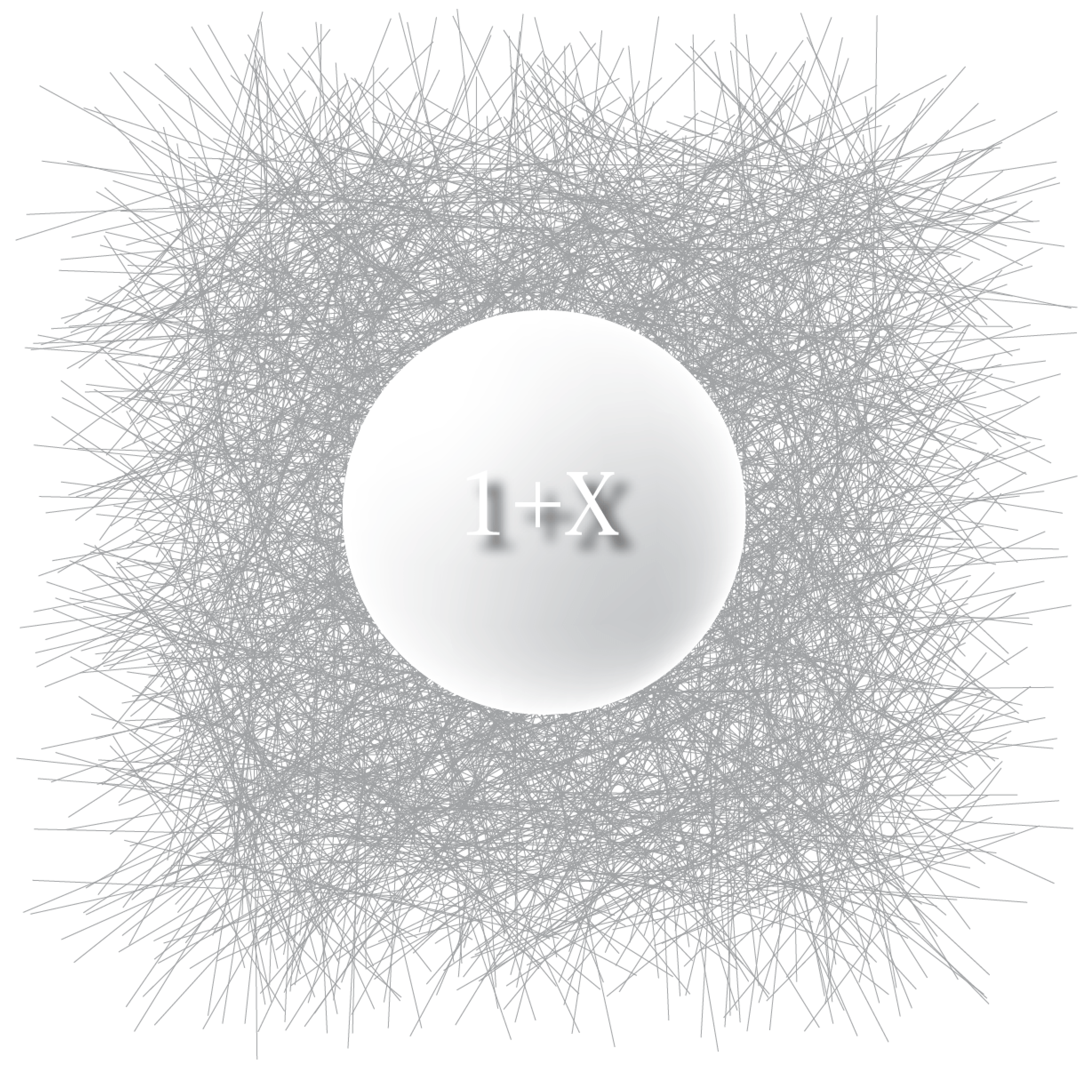
大学计算机基础教育规划教材

Web标准网页设计与PHP

唐四薪 谭晓兰 谢海波 编著



清华大学出版社

大学计算机基础教育规划教材

Web标准网页设计与PHP

唐四薪　谭晓兰　谢海波　编著

清华大学出版社

北　京

内 容 简 介

本书全面介绍了Web标准网页设计与PHP动态网页技术，采用"原理＋实例＋综合案例"的编排方式，在叙述有关原理时安排了大量的相关实例，使读者能迅速理解有关原理的用途。本书分为10章，内容包括网页与网站的基础知识，HTML、CSS、JavaScript等前端网页开发技术和PHP后台程序设计。全书遵循Web标准，面向工程实际，强调原理性与实用性。

本书可作为高等院校各专业"网页设计与制作"课程的教材，也可作为网页设计、网站制作的培训类教材，还可供网页设计和开发人员参考使用。

图书在版编目(CIP)数据

Web标准网页设计与PHP/唐四薪，谭晓兰，谢海波编著. --北京：清华大学出版社，2016
大学计算机基础教育规划教材
ISBN 978-7-302-44144-1

Ⅰ. ①W…　Ⅱ. ①唐…　②谭…　③谢…　Ⅲ. ①网页制作工具—程序设计—高等学校—教材　②PHP语言—程序设计—高等学校—教材　Ⅳ. ①TP393.092　②TP312

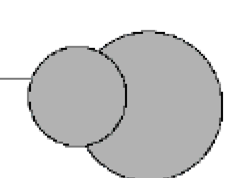
中国版本图书馆CIP数据核字(2016)第148556号

责任编辑：张　民　李　晔
封面设计：傅瑞学
责任校对：时翠兰
责任印制：何　芊

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn，010-62795954
印 装 者：北京鑫海金澳胶印有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：24.5　　**字　　数**：566千字
版　　次：2016年10月第1版　　**印　　次**：2016年10月第1次印刷
印　　数：1～2000
定　　价：49.50元

产品编号：070081-01

前　言



网页设计技术在最近10余年的发展过程中，已经发生了很多革命性的变化，如"Web标准"从提出到被普遍接受，HTML 5逐渐被浏览器支持并在移动开发中广泛应用。目前网页设计工程领域招聘网页设计人员时最常见的要求就是要懂DIV+CSS，并能够手工编写代码制作网页。这些要求代表了网页设计技术的发展趋势。

网站开发的另一方面是网站的程序设计，PHP作为网站程序设计的流行技术，在国内外的应用发展非常迅速，许多大型的网站（如淘宝、搜狐等）都采用PHP作为网站的开发工具；同时，通过对众多软件企业的调查，可以发现各种企业对于PHP开发人才的需求缺口很大。与此不相称的是，PHP在我国高校教学中还未受到足够重视。虽然很多专业都已开设了Web编程方面的课程，但是该门课程的内容以讲述ASP.NET、ASP或JSP语言为主，可见PHP尚未在高校教学中取得足够的重视，但PHP的培训课程却在大量培训机构中广泛开设。

本书在编排时考虑到网站开发技术的系统性和高校的教学需要。由于Web标准仅仅涉及网页的前端开发技术，主要是XHTML和CSS，但很多专业在开设网站开发类课程时，授课的内容大多会包括静态网页和动态网站技术两方面。因此本书还包括了JavaScript和PHP的内容，PHP作为动态网页的经典技术，具有简单易学、实验环境容易配置等优点。通过学习PHP，能为将来学习其他动态网页技术打下良好的基础。本书对于PHP程序实例，在其静态网页设计部分仍然遵循Web标准，采用DIV+CSS布局。而JavaScript作为Web前端开发技术已越来越受到追求用户体验的互联网企业的重视，本书介绍了JavaScript的入门知识和关键技术。

网页设计这门课程的特点是入门比较简单，但它的知识结构庞杂，想要成为一名有用的网页设计师是需要较长时间的理论学习和大量的实践操作及项目实训的。学习网页设计有两点是最重要的：一是务必要重视对原理的掌握；二是在理解原理的基础上一定要多练习，多实践，通过练习和实践总能发现很多实际的问题。本书在编写过程中注重"原理"和"实用"，这表现在所有的实例中都是按照其涉及的原理分类，而不是按照应用的领域分类，将这些实例编排在原理讲解之后，就能使读者迅速理解原理的用途，同时由于加深了对原理的理解，可以对实例举一反三。

在测试网页时，一定要使用不同的浏览器进行测试，建议读者至少在计算机上安装IE和Firefox（或Chrome）两种浏览器，这不仅因为制作出各种浏览器兼容的网页是网页设计的一项基本要求，更重要的是通过分析不同浏览器的显示效果可以对网页设计的各种原理有更深入的理解。

本书的内容包含了Web开发技术的各个方面，如果要将整本书的内容讲授完毕，大约需要90学时。考虑到“因材施教”的目的，本书的部分内容（在节名后注有“*”号）主要供学有余力的学生自学。

本书由唐四薪、谭晓兰、谢海波编著，唐四薪编写了第4～10章的内容，谭晓兰编写了第1章的内容，谢海波编写了第2章的部分内容。参加编写的还有刘辉、陆彩琴、喻缘、康江林、袁建君、刘艳波、舒清健、尹军、刘燕群、唐沪湘、邓明亮和唐金娟等。本书系湖南省教育科学“十二五”规划课题“异构环境下个性化网络学习资源的提取与聚合研究”（XJK013BXX005）的研究成果。

由于编者水平和教学经验有限，书中错误和不妥之处在所难免，欢迎广大读者和同行批评指正。

编　者

2016年4月

目　录

第1章 网页设计概述

Internet 是全球范围的互联网，用来实现资源共享与通信联络。Internet 可提供很多种服务，如信息浏览、电子邮件、文件传输、即时通信等。

WWW（World Wide Web，简称为 Web 或“万维网”）是 Internet 实现信息资源共享的主要途径。WWW 由遍布在 Internet 中的被称为 Web 服务器的计算机组成，Web 服务器是 Internet 上一台具有独立 IP 地址的计算机，用来存储和发布网页。接入 Internet 的计算机能够从 Web 服务器上获取信息或发送信息给 Web 服务器，其本质是通过 HTTP 协议传输基于超文本（Hypertext）的信息。

WWW 是由无数个网站连接而成的页面式网络信息系统，通过浏览器（Browser）提供一种友好的信息查询界面（网页）供用户浏览查询。WWW 之所以能够流行起来，在于它有以下几个特点：首先，WWW 具有图形化的浏览和操作界面，通过超文本可以将图形、音频、视频信息集成于同一个页面中；其次，WWW 是非常易于导航的，只要单击超链接就可以在各页面、各站点之间自由浏览；再次，WWW 与平台无关，无论是 Windows、Linux 还是苹果系统，都可以访问 WWW 上的任何信息。

1.1 初识网页设计

如果将 WWW 看成是 Internet 上的一座大型图书馆，网站就像图书馆中的一本书，而网页便是书中的一页。

1.1.1 什么是网页

一个网页就是一个文件，存放在世界某处的一台 Web 服务器中。

当用户在浏览器地址栏输入网址后，经过 HTTP 协议的传输，网页就会被传送到用户的计算机中，然后通过浏览器解释网页的内容，再展示到用户的眼前。图 1-1 是 IE 浏览器打开的网页。在浏览器窗口中，右击，执行“查看源文件”命令，就会打开一个纯文本文件，如图 1-2 所示。这个文本文件中的内容叫作 HTML 代码，浏览器的本质其实是把 HTML 代码解释成用户所看到的网页的工具。

HTML 是 HyperText Markup Language 的缩写，直译为“超文本标记语言”。



图 1-1　IE 浏览器中的网页

```
ec.hynu[1] - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1
-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=gb2312" />
<title>《网页设计与制作》学习网</title>
<style type="text/css">
<!--
body{
        margin: 0px auto;
        width:768px;
        background:#fff url(xxwlzx/body_bg.gif) repeat-x;
}
td {
        font-size: 12px;
        color: #333333;
}
```

图 1-2　网页的源文件

网页是用 HTML 编写的一种纯文本文件。用户通过浏览器所看到的包含了文字、图像、链接、动画和声音等多媒体信息的每个网页，其实质是浏览器对 HTML 代码进行了解释，并引用相应的图像、动画等资源文件，才生成了多姿多彩的网页。

但是，一个网页并不是一个单独的纯文本文件，网页中显示的图片、动画等文件都是单独存放的，以方便多个网页引用同一张图片，这和 Word 等格式的文件有明显区别。

1.1.2　网页设计的两个基本问题

网页设计是艺术与技术的结合。从艺术的角度看，网页设计的本质是一种平面设计，就像出黑板报、设计书的封面等平面设计一样，对于平面设计我们需要考虑两个基本问

题，那就是布局和配色。

1. 布局

对于一般的平面设计来说，布局就是将有限的视觉元素进行有机的排列组合，将理性思维个性化地表现出来。网页设计和其他形式的平面设计相比，有相似之处，它也要考虑网页的版式设计问题，如采用何种形式的版式布局。与一般平面设计不同的是，在将网页效果图绘制出来以后，还需要用技术手段（代码）实现效果图中的布局，将网页效果图转化成真实的网页。

将网页的版式和网页效果图设计出来后，就可用以下方式实现网页的布局。

（1）表格布局：将网页元素装填入表格内实现布局；表格相当于网页的骨架，因此表格布局的步骤是先画表格，再往表格的各个单元格中填内容，这些内容可以是文字或图片等一切网页元素。

（2）DIV＋CSS 布局：这种布局形式不需要额外的表格做网页的骨架，它是利用网页中每个元素自身具有的“盒子”来布局，通过对元素的盒子进行不同的排列和嵌套，使这些盒子在网页上以合适的方式排列就实现了网页的布局。在网页布局技术的发展过程中，产生了 Web 标准的讨论，Web 标准倡导使用 DIV＋CSS 来布局。

（3）框架布局：将浏览器窗口分割成几部分，每部分放一个不同的网页，这是很古老的一种布局方式，现在用得较少。

网页设计从技术角度看，就是要运用各种语言和工具解决网页布局和美观的问题，所以网页设计中很多技术都仅仅是为了使网页看起来更美观。常常会为了网页中一些细节效果的改善，而花费大量的工作量，这体现了网页设计师追求完美的精神。

2. 配色

网页的色彩是树立网页形象的关键要素之一。对于一个网页设计作品，浏览者首先看到的不是图像和文字，而是色彩搭配，在看到色彩的一瞬间，浏览者对网页的整体印象就确定下来了，因此说色彩决定印象。一个成功的网页作品，其色彩搭配可能给人的感觉是自然、洒脱的，看起来只是很随意的几种颜色搭配在一起，其实是经过了设计师的深思熟虑和巧妙构思的。

对于初学者来说，在用色上切忌将所有的颜色都用到，尽量控制在三种色彩以内，并且这些色彩的搭配应协调。而且一般不要用纯色，灰色适合与任何颜色搭配。

1.1.3 网页设计语言——HTML 简介

网页是用 HTML 语言编写的。HTML 作为一种建立网页文档的语言，它用标记标明文档中的文本及图像等各种元素，指示浏览器如何显示这些元素。HTML 具有语言的一般特征。所谓语言，是一种符号系统，具有自己的词汇（符号）和语法（规则）。

所谓标记，就是作记号。例如，为了让浏览器理解某段内容的含义，HTML 将各种内容写在标记内，以标明其含义。例如：

```
<标记>受标记影响的内容</标记>
```

这就和我们写文章时用粗体字表示文章的标题、用换行空两格表示一个段落类似，HTML 是用一对<h1>标记把文字括起来表示这些字是一级标题，用<p>标记把一段字括起来表示这是一个段落。

所谓超文本，就是相比普通文本有超越的地方，如超文本可以通过超链接转到指定的某一页，而普通文本只能一页页翻。超文本还具有图像、视频、声音等元素，这些都是普通文本所无法具有的。

HTML 的发展历程如图 1-3 所示，它是 SGML（标准通用标记语言）在 WWW 中的应用。1999 年 HTML 4.01 发布，是 HTML 最成熟的一个版本。HTML 4.01 后的一个修订版本是 XHTML 1.0，该版本并没有引入新的标记或属性，唯一的区别只是语法更加严格，但 XHTML 于 2009 年被 W3C 放弃，转而发布了新的 HTML 5 标准。目前一些新版本的浏览器已开始支持 HTML 5，但 IE 10 以下版本的浏览器对 HTML 5 的支持仍很不好。

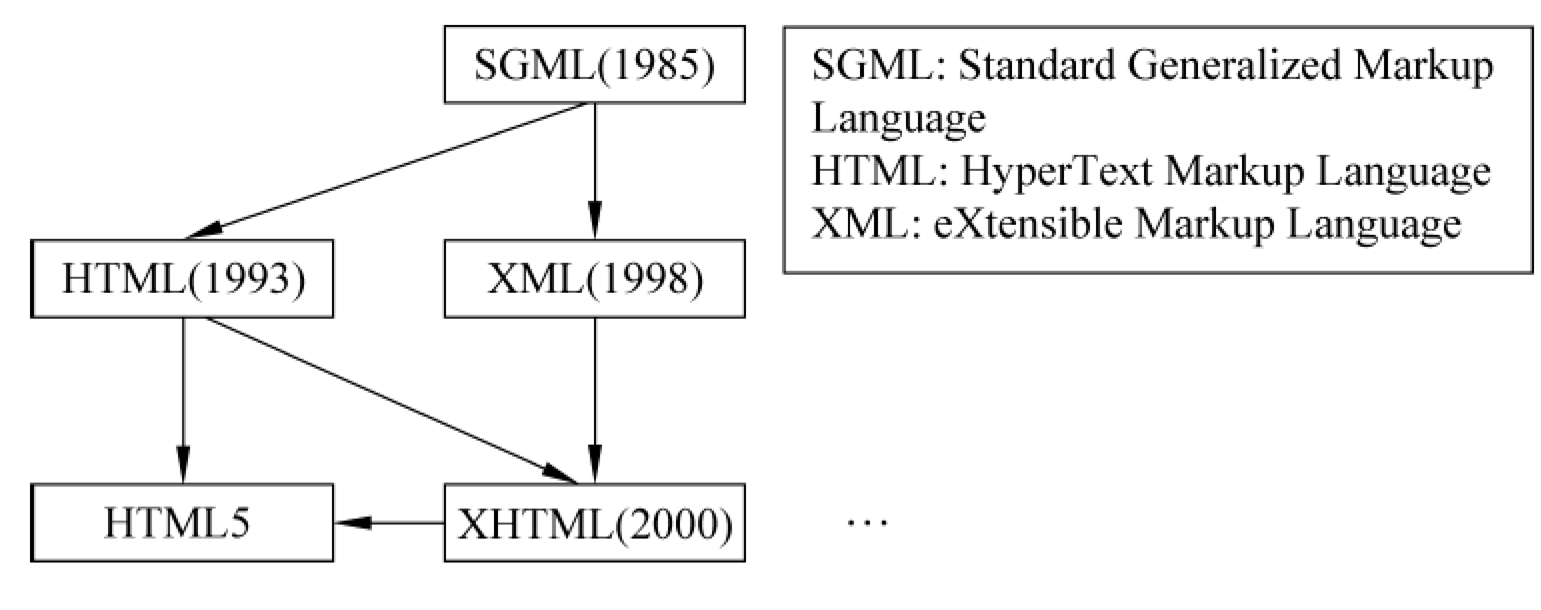


图 1-3　标记语言的发展过程

HTML 语言与编程语言有明显不同，首先它不是一种计算机编程语言，而是一种描述文档结构的语言，或者说是排版语言；其次，HTML 是弱语法语言，随便怎么写都可以，浏览器会尽力去理解执行，不理解的按原样显示，而编程语言是严格语法的语言，写错一点点计算机就不执行，报告错误；最后，HTML 语言不像大多数编程语言那样需要编译成指令后执行，而是每次由浏览器解释执行。

1.1.4　网页制作软件

网页的本质是纯文本文件，因此可以用任何文本编辑器制作网页，但这样必须完全手工书写 HTML 代码。为了提高网页制作的效率，人们一般借助于专业的网页开发软件制作网页，它们具有“所见即所得”（what you see is what you get）的特点，可以不用手工书写代码，通过图形化操作界面就能插入各种网页元素，如图像、表格、超链接等，而且能在设计视图中实时看到网页的大致浏览器效果。目前流行的专业网页开发工具主要有：

（1）Adobe 公司的 Dreamweaver CS6。Dreamweaver（本书简称 DW）的中文含义是“织梦者”，DW 具有操作简捷、容易上手等优点，是目前最流行的网页制作软件。

由于 Dreamweaver CS 以上版本对中文的支持不太好，且设置站点和预览网页时选项过多，建议初学者可以使用经典版本 Dreamweaver 8 进行网页设计学习。

（2）Microsoft 的 Expression Web 4。它是微软开发的新一代网页制作软件，对 Web

标准的内建支持，使其能帮助开发者建立跨浏览器兼容性的网站。

虽然这些软件都具有“所见即所得”的网页制作能力，可以让不懂 HTML 语言的用户也能制作出网页。但如果想灵活制作精美专业的网页，很多时候还是需要在代码视图中手工修改代码，因此学习网页的代码对网页制作水平的提高是很重要的。

DW 等软件同时具有很好的代码提示和代码标注功能，使得手工编写和修改代码也很容易，并且还能报告代码错误，所以就算是手工编写代码，也推荐使用这些软件。

DW 和 Expression Web 同时具有强大的站点管理功能，因此它们还是网站开发工具，网页制作和网站管理的功能被集成于一体。

1.2　网站的创建和制作流程

网站就是由许多网页及资源文件（如图片）组成的一个集合，网页是构成网站的基本元素。通常把网站内的所有文件都放在一个文件夹中，所以网站从形式上看就是一个文件夹。

设计良好的网站通常是将网页文件及其他资源文件分门别类地保存在相应的目录中，以方便管理和维护，这些网页通过超链接组织在一起，图 1-4 是一个网站目录示意图。用户浏览网站时，看到的第一个网页称为主页（也叫首页），上面通常设有网站导航，链接到站内各主要网页，其名称一般是 index.htm 或 index.html，必须放在网站的根目录下，即网站目录是首页文件的直接上级目录。一个网站对应一个网站目录，所以网站目录是唯一的。

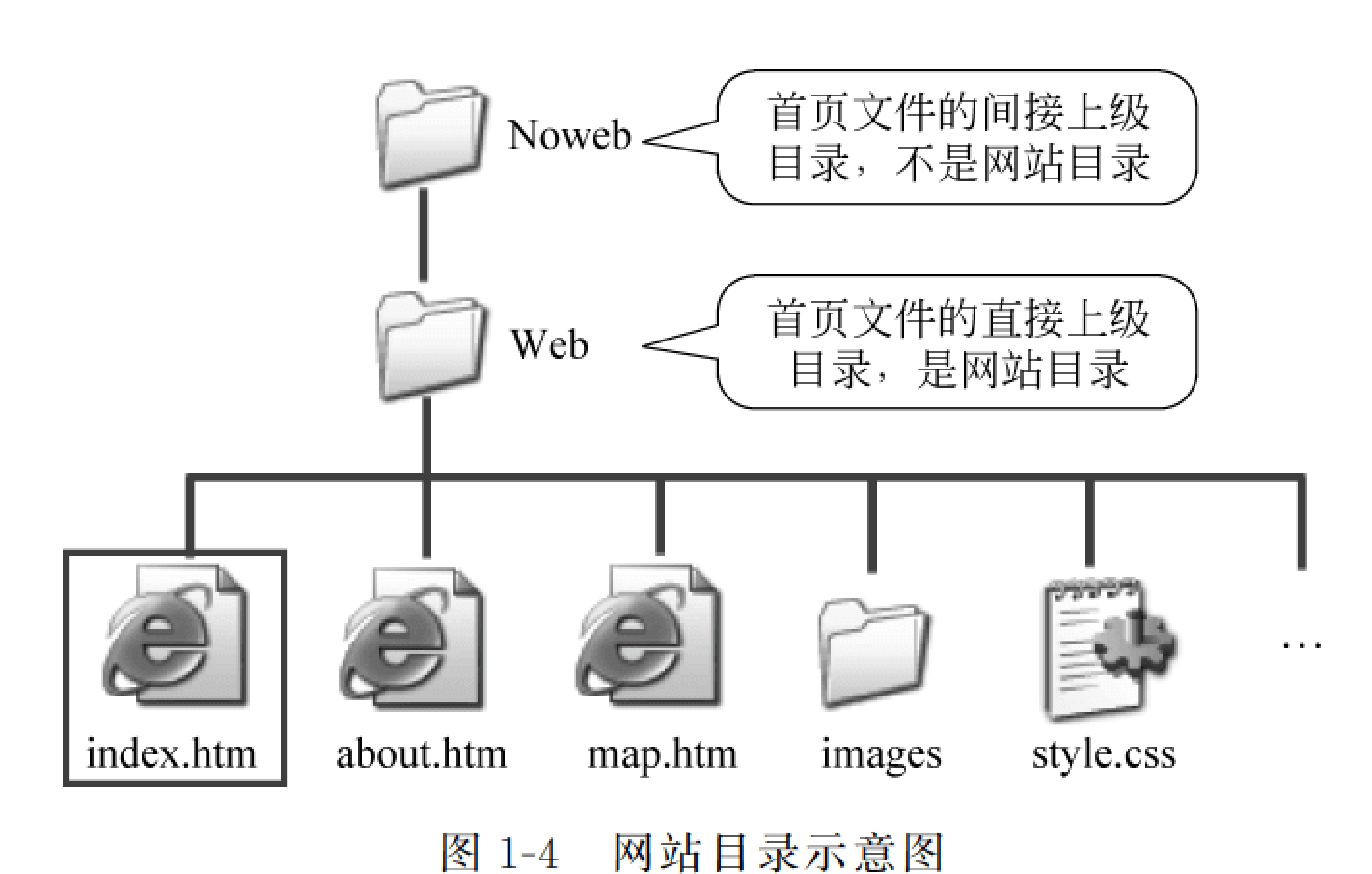


图 1-4　网站目录示意图

通常在网站目录下新建一个名为 images 的子目录，用于保存网站中所有网页需要调用的图片文件。

因此，制作网站的第一步是在硬盘上新建一个目录（如 D:\web），作为网站目录。网站制作完成后将这个目录中的所有内容上传到远程服务器中就可以了。

1.2.1　网站的特征

从用户的角度看，设计良好的网站一般具有如下几个特征：

(1) 拥有众多的网页。从某种意义上说,建设网站就是制作网页,网站主页是最重要的网页。

(2) 拥有一个主题与统一的风格。网站虽然有许多网页,但作为一个整体,它必须有一个主题和统一的风格。所有的内容都要围绕这个主题展开,和主题无关的内容不应出现在网站中。网站内所有网页要有统一的风格,主页是网站的首页,也是网站最为重要的网页,所以主页的风格往往决定了整个网站的风格。

(3) 有便捷的导航系统。导航是网站非常重要的组成部分,也是衡量网站是否优秀的一个重要标准。设计良好的网站都具有便捷的导航工具,可以帮助用户以最快的速度找到自己需要的网页。导航系统常用的实现方法有导航条、路径导航、链接导航等。

(4) 分层的栏目组织。将网站的内容分成若干个大栏目,每个大栏目的内容都放置在网站目录下的一个子目录中,还可将大栏目再分成若干小栏目,也可将小栏目分成若干个更小的栏目,都分门别类放在相应的子目录下,这就是网站采用的最简单的层次型组织结构,结构清晰的网站可大大方便网站的维护和管理。

1.2.2 网站的开发步骤

网站开发也可看成是一种软件开发,其开发过程可分为前期工作、中期工作和后期工作。尽管本书介绍的内容主要是中期工作,但网站开发的前期工作对网站开发的成功与否起着非常关键的作用。

1. 网站需求分析与定位

假设为一家单位或部门制作网站,则需求分析首先是需要了解该单位的主要职能。然后结合该单位的实际工作需求,并对需求进行明确和细化,确定网站应满足的设计目标和要求。这一步的主要任务是采集和提炼用户需求。其关键是:

(1) 明确目标,弄清楚用户的真正需求;

(2) 有效沟通,通常用户的需求是凌乱的、不完整的,很多用户知道自己想要什么,但是表达不出来。这需要网站需求分析人员深入企业内部,熟悉用户的业务流程。

2. 确定网站内容、风格和功能

网站内容主要来源于建站客户提供的资料,若客户提供的资料较少,可以参考同类网站,适当丰富一些内容。若客户提供的资料很多,则要分清主次,整理分类,不要全部放在网站上。网站内容可以是文字、图表、图片、动画、视频等多种格式。

网站风格应以网站性质和网站内容为基础,例如,如果是企事业单位的门户网站,则网站风格应以简洁端庄为宜;如果是娱乐、旅游网站,则网站风格可偏向活泼。网站风格成功的关键是避免雷同和过分修饰,即使是同一类型的网站,也应该有自己的特点、自己的风格。

根据用户的需求确定网站的功能,如企业网站应具备的基本功能一般有:信息发布与维护、信息查询、留言本、网上订货、在线招聘等。

3. 规划网站栏目

根据企事业单位业务的侧重点，结合网站定位来确定网站的栏目。开始时可能会因为栏目较多而难以确定最终需要的栏目，这就需要展开另一轮讨论，需要网站设计人员和客户在一起阐述自己的意见，一起反复比较，将确定下来的内容进行归类，从而确定网站的一级栏目，然后对每个栏目中的内容进行细化确定二级栏目（甚至三级栏目）等。最终形成网站栏目的树状结构，用来清晰表达站点结构。

4. 设计网页效果图和切图

网页效果图在设计之前应先用铅笔勾勒出一个草图，草图主要是对版面进行划分，确定各个模块在网页中的大体位置。接下来就是使用 Photoshop 或 Fireworks 等图像处理软件绘制网页效果图，这一步的核心任务是美术设计，通俗地说，就是让页面更美观、更漂亮。在一些比较大的网页开发项目中，通常都会有专业的美工参与，这一步就是美工的任务。

通常一个网站中有成百上千个页面，但实际上不必为每个页面都设计网页效果图，因为网站中的页面一般分为三类，即首页、栏目首页和内页。相同类型的页面，其布局和外观都是相同的，只是里面的文字信息不同而已。因此制作网页效果图只需要分别制作首页、栏目首页和内页三个页面的效果图。我们一般都是先制作首页的效果图，因为首页的效果图是最复杂的，然后再对首页效果图的中间区域进行修改，制作出栏目首页和内页的效果图。如果要制作手机版网站，还需要另外设计手机网站的首页、栏目首页和内页。

为了让网站有整体感，应该在所有页面中放置一些贯穿性的元素，即在网站的所有页面中都出现的元素。例如，网站中所有网页的头部、导航条和页尾都是相同的。

网页效果图应该按照真实网页大小 1∶1 的比例制作，并且尽量提供 Fireworks 或 Photoshop 中图层未合并的原始文件。这样在将效果图转换成真实的网页时才能方便地获得背景图案、各个模块的精确大小、在页面中的位置及文本的字体大小、字体类型等信息。

网页效果图制作完成后，还必须对效果图进行切片，因为真实的网页并非一张图片，而是将许多小图片放置在网页合适的位置上。网页切片就是根据网页的需要，将有用的图片从网页效果图中切出来，其他不需要的图片（如文字块区域、单色背景区域）则可以舍去，然后再在对应的区域填充文字。网页切片需要考虑网页布局采用的技术。使用表格布局的网页和 CSS 布局的网页其需要的图片往往是不相同的，因此切片方式也不相同。

提示：虽然有些简单的网页也可以不绘制网页效果图，而是分别制作网页中各部分的图片，再把它们插入网页中对应的位置上。但这样做的缺点也是相当明显的：由于各个位置的图片是单独绘制的，容易造成网页整体风格不统一，而且网页各个区域也显得是孤立存在的，过渡不自然。因此要制作出看起来浑然一体高水平的网页作品，绘制网页效果图是不可省略的步骤。

5. 制作静态模板页面

网页效果图设计完成后,接下来就是根据网页效果图设计网站的静态模板页面,这一步的要求是制作出来的网页要和网页效果图尽量完全相似。网页制作人员只有熟练掌握网页布局的技巧才能完成。

静态模板页面的制作过程也应遵循先首页、再栏目首页、最后内页的过程。这是因为首页最复杂,并且首页中的很多网页元素可以提供给栏目首页和内页重用。为了测试静态页面的整体效果,可以在页面中输入一些无关文字。在将页面转换为动态页面时,再将这些静态文字转换为动态字段即可。

6. 绑定动态数据和实现后台功能

目前的网站一般都是将信息保存在数据库中的动态网站,这样能方便地添加、删除和修改网站中的信息。这一步的任务就是将静态模板页转换为动态网站。为此,需要由程序员根据功能需求来编写网站的后台管理程序。但由于完全自己编写后台程序的工作量很大,现在更流行调用一个通用的后台管理系统(也称为 CMS(Content Management System),内容管理系统),这样开发程序的工作量就小多了。

调用后台管理系统的步骤一般是:首先在后台管理系统中添加几条新闻记录;然后打开后台管理系统的数据库查看标题、内容等字段的字段名;最后在前台的静态页中连接数据库并绑定相关内容对应的字段名,这称为绑定动态数据,这样前台页面就可以显示数据库中的内容了,而后台管理系统也可以对这些内容进行添加、删除和修改。

7. 整合与测试网站

当制作和编程工作都完成以后,就需要把实现各种功能的程序和页面进行整合。整合完成后,需要进行内部测试,测试成功后即可将网站目录下的所有文件上传到服务器上,上传服务器一般采用 FTP 文件传输协议上传。

例如,假设远程服务器的 FTP 地址是 ftp://011. seavip. cn,则在浏览器(或 Windows 资源管理器)的地址栏中输入 FTP 地址,这时会弹出“登录身份”对话框要求输入用户名和密码,输入正确后,就会显示一个类似于资源管理器的界面,表示远程服务器上的网站目录,把本机网站目录中的文件通过拖动复制到该窗口中的目录下就实现了文件上传,还可以对远程网站目录中的文件或文件夹进行删除、新建等操作。

1.2.3 在 Dreamweaver 中建立站点

在 Dreamweaver 中,“站点”一词既表示网站,又表示该网站对应文件夹的本地存储位置。在开始建立 Web 站点之前,我们需要建立站点文档在本地的存储位置(即网站目录)。Dreamweaver 站点可组织与 Web 站点相关的所有文档,跟踪和维护链接、管理文件、共享文件以及将站点文件传输到 Web 服务器上。

要制作一个能够被访问者浏览的网站,首先需要在本机上制作网站,然后上传到远程服务器。放置在本地磁盘上的网站目录被称为“本地站点”,传输到远程 Web 服务器

中的网站被称为“远程站点”。Dreamweaver 提供了对本地站点和远程站点强大的管理功能。

因而应用 Dreamweaver 不仅可以创建单独的网页，还可以创建完整的网站。使用 Dreamweaver 建立网站时，首先必须告诉 Dreamweaver 本地站点对应于哪个硬盘目录，即把我们在硬盘上建立的网站主目录定义为 Dreamweaver 的站点。因此在 Dreamweaver 下新建站点是用 Dreamweaver 开发网站的第一步。

提示：虽然不新建站点也可以使用 Dreamweaver 编辑单个网页，但是强烈建议初学者在制作网站之前一定要先新建站点，因为新建站点之后：

（1）在网页之间建立超链接时，Dreamweaver 能使站点内的网页以相对 URL 的方式进行链接，这种形式的超链接代码在上传到服务器后也无须做任何更改；

（2）新建网页时所有的网页文件都会自动保存在站点目录中，便于管理；

（3）在预览动态网页时，Dreamweaver 还能使用已设置好的 URL 运行该动态网页。

下面介绍在 Dreamweaver 中新建站点的步骤。

（1）启动 Dreamweaver，在 Dreamweaver 中执行“站点”→“新建站点”菜单命令，就会弹出新建站点对话框。这个对话框分为“基本”和“高级”两个选项卡，“基本”选项卡可分步骤完成一个站点的建立，“高级”选项卡则是用来直接设置站点的各个属性。在此“基本”选项卡中输入站点的名称（可任取一个站点名，如 hynu），对于静态站点，HTTP 地址不需要设置，如图 1-5 所示。



图 1-5 新建站点对话框(一)

（2）单击“下一步”按钮，在弹出的如图 1-6 所示的对话框中选择“否，我不想使用服务器技术。”按钮，此对话框用于设置站点文件类型。如果要制作动态网页，则应选择“是，我想使用服务器技术。”按钮，此时将出现一个选择具体动态网页技术的下拉列表框，可选择需要的动态网页技术。

（3）单击“下一步”按钮，弹出如图 1-7 所示的对话框，因为通常我们都是在本地机器上做好网站，再上传到远程的服务器去，所以选择“编辑我的计算机上的本地副本，完成后再上传到服务器(推荐)”这一默认选项。



图 1-6 新建站点对话框(二)

对于选项“您将把文件存储在计算机上的什么位置?”，显然，我们应该把制作的网页保存在网站主目录中。因此，这里应该输入(或选择)网站主目录的路径。

当把已经新建好的网站主目录作为 Dreamweaver 的站点后，Dreamweaver 就会把以后新建的网页文件默认都保存在该站点目录中，并且站点目录内，网页之间的链接会使用相对链接的方式。

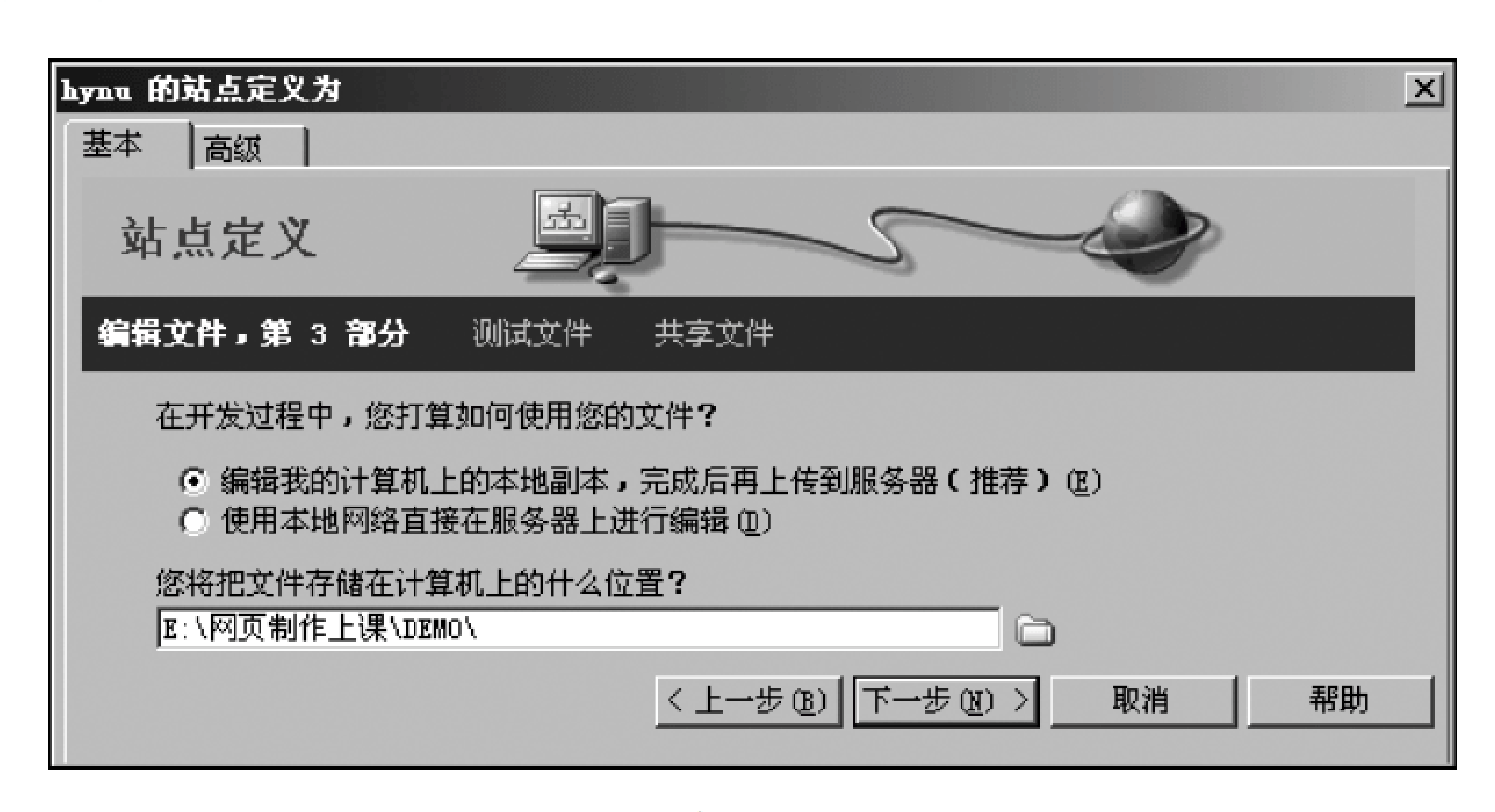


图 1-7 新建站点对话框(三)

提示：对网站目录和网页文件命名应避免使用中文，尤其对于动态网页或将网页上传到服务器后，使用中文很容易出问题。例如图 1-7 中站点目录“DEMO”就不是中文。

(4) 单击“下一步”按钮，在“您如何连接到远程服务器?”下拉列表框中选择“无”选项。

(5) 单击“下一步”按钮，弹出站点信息总结的对话框，单击“完成”按钮就完成了一个本地站点的定义。

(6) 定义好本地站点之后，Dreamweaver 窗口右侧的“文件”面板(见图 1-8)就会显示刚才定义的站点的目录结构，可以在此面板中右击，通过快捷菜单命令在站点目录内新建文件或子目录，这与通过资源管理器在网站目录中新建文件或目录的效果是一样的。



图 1-8 “文件”面板

如果要修改定义好的站点，只需执行“站点→管理站点”菜

单命令,选中要修改的站点名,单击“编辑”按钮,就可在站点定义对话框中对原来的设置进行修改。

1.3　Web 服务器与浏览器

在学习网页制作之前,有必要了解“浏览器”和“服务器”的概念。网站浏览者坐在计算机前浏览各个网站上的内容,实质上是从远程的计算机中读取了一些内容,然后在本地的计算机中显示出来的过程。提供内容信息的远程计算机称为“服务器”,浏览者使用的本地计算机称为“客户端”,客户端使用“浏览器”程序,就可以通过网络接收到“服务器”上的网页以及其他文件,因此我们浏览的网页是保存在服务器上的,服务器可以同时供许多不同的客户端访问。

1.3.1　Web 服务器的作用

访问网页具体的过程是:当用户的计算机连入互联网后,通过在浏览器中输入网址发出访问某个网站的请求,然后这个网站的服务器就把用户请求的网页文件传送到用户的浏览器中,即将文件下载到用户计算机中,浏览器再解析并显示网页文件内容,这个过程如图 1-9 所示。

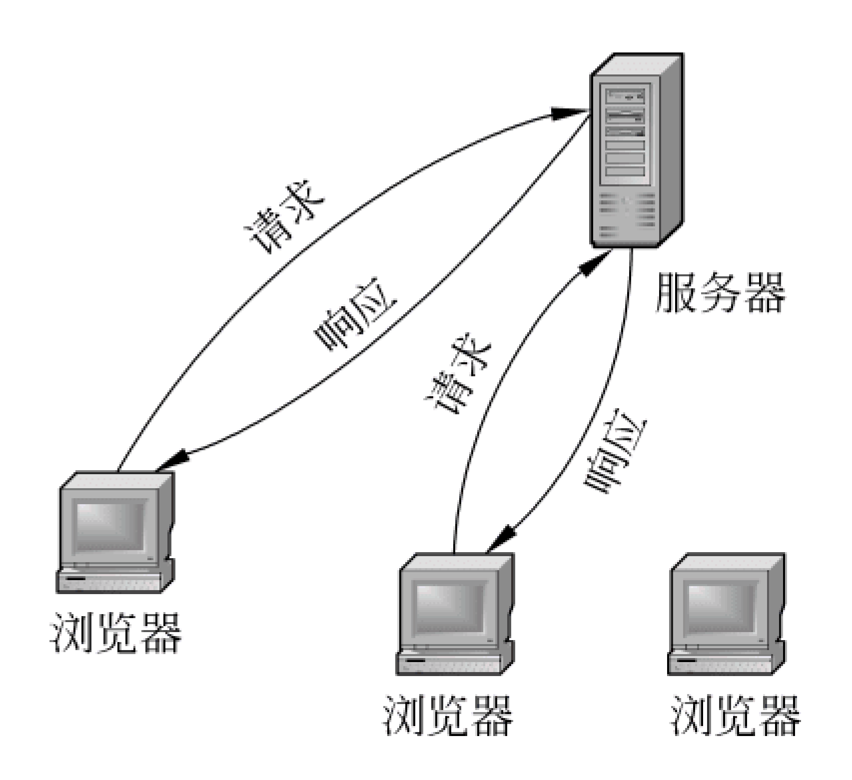


图 1-9　服务器与浏览器之间的关系

对于静态网页(不含有服务器端代码,不需要 Web 服务器解释执行的网页)来说, Web 服务器只是到服务器的硬盘中找到该网页并发送给用户计算机,起到的只是查找和传输文件的作用。因此在测试静态网页时可不安装 Web 服务器,因为制作网页时网页还保存在本地计算机中,可以手工找到该网页所在的目录,双击网页文件就能用浏览器打开它。

1.3.2　浏览器的种类和作用

浏览器是供用户浏览网页的软件,其功能是读取 HTML 等网页代码并进行解释以生成用户看到的网页。

1. 浏览器的种类

浏览器的种类很多,目前常见的浏览器有:IE 6～IE 11、Firefox 30、Google Chrome 2、Safari 5、Opera 10 等。图 1-10 是各种浏览器的徽标。

(1) IE(Internet Explorer)是 Windows 操作系统自带的浏览器,也是国内使用最广泛的浏览器。目前常用的 IE 浏览器版本很多,从 IE 6 到 IE 11 都有,各种版本的 IE 浏览器对网页的解析区别又很大。其中 IE 6 对 Web 标准的支持不太好,并存在一些明显的 bug,而 IE 8 开始对 Web 标准的支持得到了显著改善。从 IE 9 开始,IE 浏览器逐步支持


IE


Firefox


Opera


Safari


Google Chrome

图 1-10　各种常见的浏览器

HTML 5 和 CSS 3 版本。由于 IE 6 仍然是 Windows XP 默认的浏览器，因此在制作网页时，仍然要考虑代码在 IE 6 浏览器上的兼容性。

（2）Google Chrome 是浏览器市场的后起之秀，该浏览器具有运行速度快，占用资源少的特点，目前其市场占有率快速提高，预计将会成为一种最流行的浏览器。

（3）Firefox 是网页设计领域推荐的标准浏览器，它对 Web 标准和 CSS 有很好的支持，并且是最先支持 HTML 5 的浏览器。

（4）Safari 5 最初是苹果电脑（包括 iPad、iPhone）上的浏览器，目前 Safari 也有 Windows 版本，该浏览器解释 JavaScript 脚本的速度很快。

（5）Opera 10 是一款小巧的浏览器，在手持设备的操作系统上用得较多。

目前网页在各种浏览器中的显示效果有时还不完全相同，随着 Web 标准的推广，按照 Web 标准制作的网页在各种浏览器中的显示效果将趋于一致。

2. 浏览器的作用

浏览器最重要或者说核心的部分是 Rendering Engine，习惯上称之为浏览器内核，负责对网页语法的解释（包括 HTML、CSS、JavaScript）并显示网页。

浏览器解释网页代码的过程类似于程序编译器编译程序源代码的过程，都是通过执行代码（HTML 代码或程序代码）再生成界面（网页或应用程序界面），不同的是浏览器对 HTML 等代码是解释执行的。不同的浏览器内核对网页代码的解释并不完全相同，因此同一网页在不同内核的浏览器中的显示效果就有可能不同。作为网页制作者，应追求网页尽可能在各种浏览器中有一致的显示效果。建议测试网页时至少应将网页在 IE 8 和 Google Chrome 两种浏览器上运行一遍，以测试网页的浏览器兼容性。

1.4　URL 与域名

1. URL 的含义和格式

当用户使用浏览器访问网站时，通常需要在浏览器地址栏中输入网站地址（网址），这个地址就是 URL（Universal Resource Locator，统一资源定位器）。URL 信息会通过 HTTP 请求发送给服务器，服务器根据 URL 信息返回对应的资源文件到浏览器。

URL 是 Internet 上任何资源的标准地址，为了使人们能访问 Internet 上任意一个网

页(或其他文件),每个网站上的每个网页(或资源文件)在 Internet 上都有一个唯一的 URL 地址,通过网页的 URL,浏览器就能定位到目标网页或资源文件。就好像邮寄信件时通过地址和姓名就能让邮局定位到收信人一样。

URL 的一般格式如下,图 1-11 是一个 URL 的示例。

```
协议名://主机名[:端口号][/目录路径/文件名][#锚点名]
```

URL 协议名后必须接"://",其他各项之间用"/"隔开,例如图 1-11 中的 URL 表示信息被放在一台被称为 www 的服务器上,hynu.cn 是一个已被注册的域名,cn 表示中国。有时也把主机名和域名合称为主机名(或主机头、域名)。域名对应服务器上的网站目录(如 D:\hynu),web/201609/是服务器网站目录下的目录路径,而 first.html 是位于上述目录下的文件名,因此该 URL 能够让我们访问到这个文件。

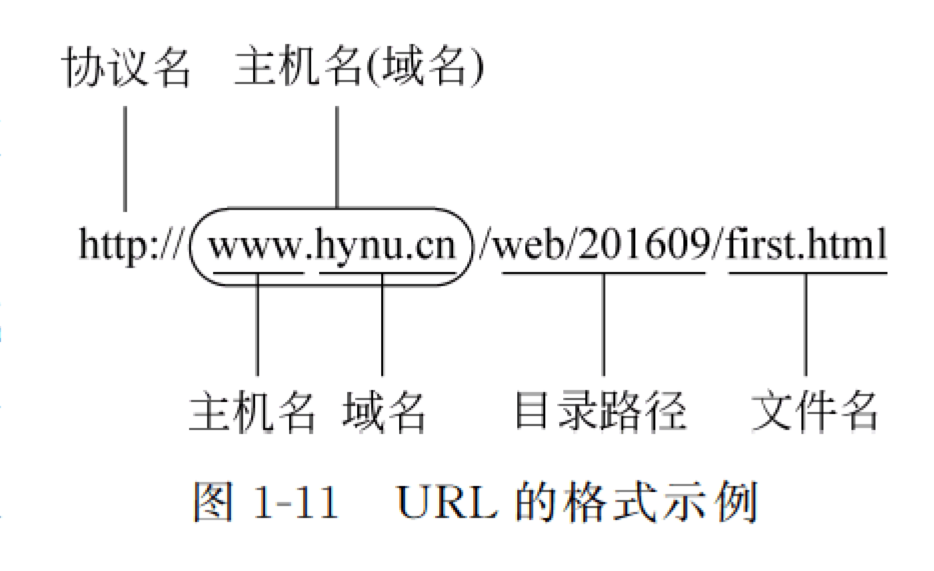


图 1-11　URL 的格式示例

在 URL 中,常见的协议名有如下三种:

(1) http——超文本传输协议,用于传送网页。

(2) ftp——文件传输协议,用于传送文件。

(3) file——访问某台主机上共享文件的协议。如果访问的是本机,则主机头可以省略,但斜杠不能省略。

下面是几个 URL 的例子。

```
http://bbs.runsky.com:8080/bbs/forumdisplay.php#fid
ftp://219.216.128.15/
file:///pub/files/foobar.txt
```

2. 域名与主机的关系

在 URL 中,主机名通常是域名或 IP 地址。最初,域名是为了方便人们记忆 IP 地址的,使用户在 URL 中可以输入域名而不必输入难记的 IP 地址。但现在多个域名可对应一个 IP 地址(一台主机),即在一台主机上可架设多个网站,这些网站的存放方式称为"虚拟主机"方式,此时由于一个 IP 地址(一台主机)对应多个网站,就不能采用输入 IP 地址的方式访问网站,而只能在 URL 中输入域名。Web 服务器为了区别用户请求的是这台主机上的哪个网站,通常必须为每个网站设置"主机头"来区别这些网站。

因此域名的作用有两个:一是将域名发送给 DNS 服务器解析得到域名对应的 IP 地址以便与该 IP 对应的服务器进行通信;二是将域名信息发送给 Web 服务器,通过域名与 Web 服务器上设置的"主机头"进行匹配确认客户端请求的是哪个网站,如图 1-12 所示。若客户端没有发送域名信息给 Web 服务器,例如直接输入 IP,则 Web 服务器将打开服务器上的默认网站。

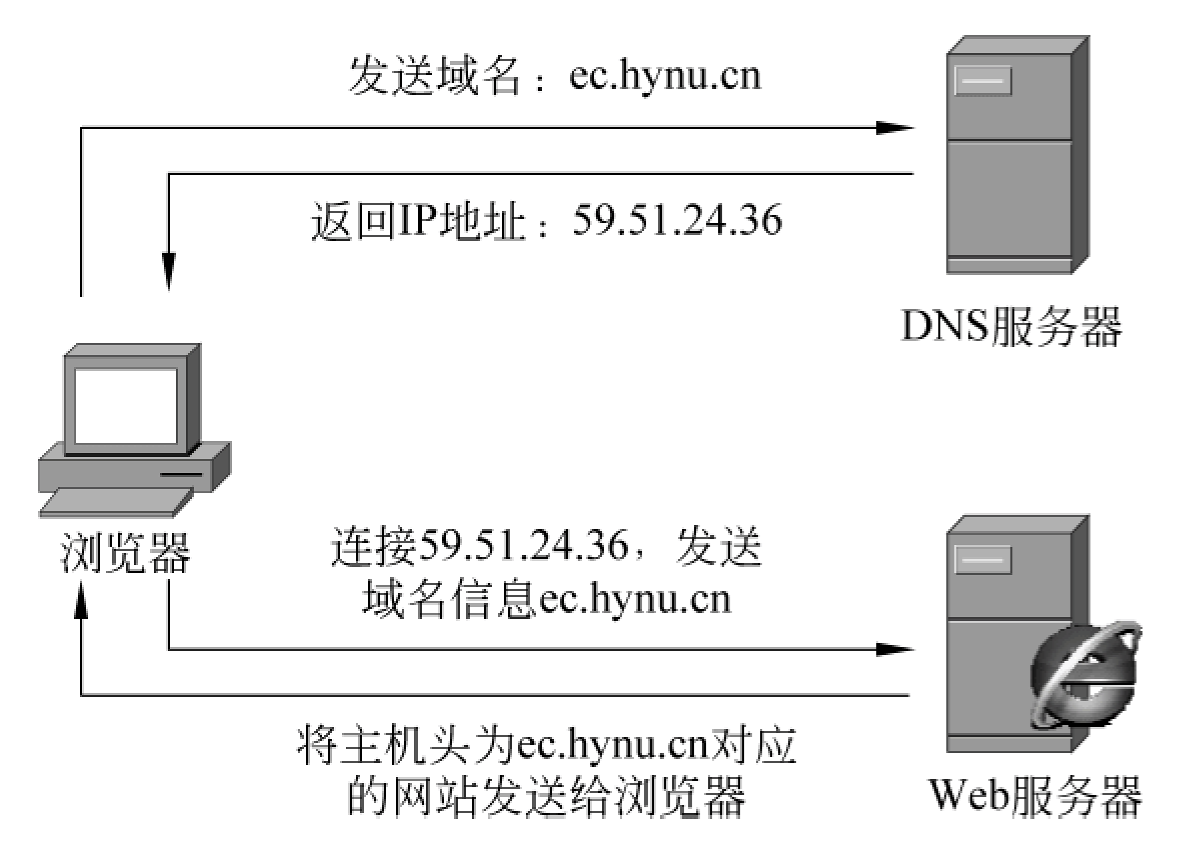


图 1-12 浏览器输入域名访问网站的过程

习 题 1

1. 对于采用虚拟主机方式的多个网站，域名和 IP 地址是(　　)的关系。

   A. 一对多　　B. 一对一　　C. 多对一　　D. 多对多

2. 网页的本质是(　　)文件。

   A. 图像　　B. 纯文本

   C. 可执行程序　　D. 图像和文本的压缩

3. 请解释 http://www.moe.gov.cn/business/moe/115078.html 的含义。

4. 简述 WWW 和 Internet 的区别。

5. 简述 URL 的含义和作用。

6. 简述网站的本质和特点。

7. 使用 Dreamweaver 新建名为“aiw”的站点，该站点对应硬盘上的“D:\aiw”网站文件夹。

8. 在计算机安装 Firefox 浏览器，并分别使用 IE 浏览器和 Firefox 浏览器查看网页的源代码。

第 2 章

HTML

网页是用 HTML(HyperText Markup Language)语言编写的,HTML 是所有网页制作技术的基础。无论是在 Web 上发布信息,还是编写可供交互的程序,都离不开 HTML 语言。

2.1　HTML 概述

HTML,即超文本标记语言。网页是用 HTML 书写的一种纯文本文件。用户通过浏览器所看到的包含了文字、图像、动画等多媒体信息的每一个网页,其实质是浏览器对该纯文本文件进行了解释,并引用相应的图像、动画等资源文件,才生成了多姿多彩的网页。

HTML 是一种标记语言。可以认为,HTML 代码就是"普通文本＋HTML 标记",而不同的 HTML 标记能表达不同的效果,如表格、图像、表单、文字等。

2.1.1　HTML 文档的结构

HTML 文件本质是一个纯文本文件,只是它的扩展名为.htm 或.html。任何纯文本编辑软件都能创建 HTML 文件。图 2-1 是 HTML 文档的基本结构。

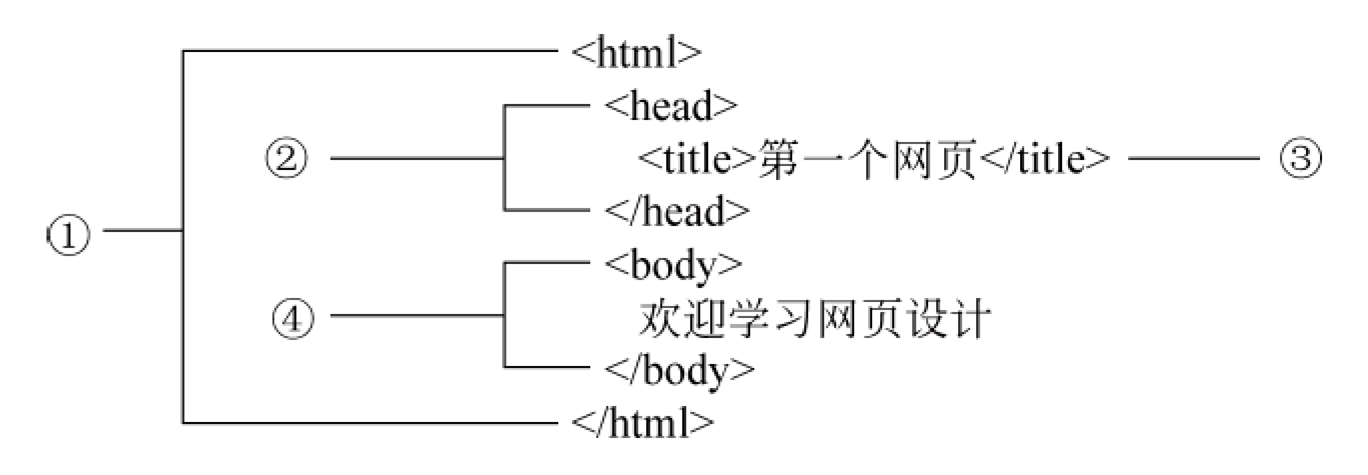


图 2-1　HTML 文档的基本结构

从图 2-1 可以看出,HTML 代码分为 3 部分,其中各部分含义如下:

(1) ＜html＞…＜/html＞:告诉浏览器 HTML 文档的开始和结束位置,HTML 文档包括 head 部分和 body 部分。HTML 文档中所有的内容都应该在这两个标记之间,一个 HTML 文档总是以＜html＞开始,以＜/html＞结束。

（2）<head>…</head>：HTML 文档的头部标记，头部主要提供文档的描述信息，head 部分的所有内容都不会显示在浏览器窗口中，在其中可以放置页面的标题<title>以及页面的类型、字符编码、链接的其他脚本或样式文件等内容。

（3）<title>…</title>：定义页面的标题，将显示在浏览器的标题栏中。

（4）<body>…</body>：用来指明文档的主体区域，主体包含 Web 浏览器页面显示的具体内容，因此网页所要显示的内容都应放在这个标记内。

提示： *HTML 标记之间只可以相互嵌套，如<head><title>…</title></head>，但绝不可以相互交错，如<head><title>…</head></title>就是绝对错误的。*

我们可以打开最简单的文本编辑器——记事本，在记事本中输入图 2-1 中的代码。输入完成后，单击“保存”菜单命令，注意先在“保存类型”中选择“所有文件”，再输入文件名为“2-1.html”。单击“保存”按钮，就新建了一个后缀名为.html 的网页文件，可以看到其文件图标为浏览器图标，双击该文件，会用浏览器显示如图 2-2 所示的网页。



图 2-2　2-1.html 在浏览器中的显示效果

2.1.2　Dreamweaver 的开发界面

Dreamweaver 为网页制作提供了简洁友好的开发环境，Dreamweaver 的工作界面包括视图窗口、属性窗口、工具栏和浮动面板组等，如图 2-3 所示。



图 2-3　Dreamweaver CS3 的工作界面

Dreamweaver 的视图窗口可在“设计视图”“代码视图”和“拆分视图”之间切换。

（1）“设计视图”的作用是帮助用户以“所见即所得”的方式编写 HTML 代码，即通过一些可视化的方式自动编写代码，减少用户手工书写代码的工作量。Dreamweaver 的设

计视图蕴含了面向对象操作的思想，它把所有的网页元素都看成是对象，在设计视图中编写 HTML 的过程就是插入网页元素，再设置网页元素的属性。

(2) “代码视图”供用户手工编写或修改代码，因为在网页制作过程中，有些操作不能(或不方便)在设计视图中完成，此时用户可单击“代码”按钮，切换到代码视图直接书写代码，代码视图拥有代码提示的功能，即使是手工编写代码，速度也很快。

(3) “拆分视图”同时显示设计视图和代码视图，在用户需要寻找代码与对应的网页区域时可切换到这种视图。

为了提高网页制作的效率，建议用户首先在“设计视图”中插入主要的 HTML 元素(尤其是像列表、表格或表单等复杂的元素)，然后切换到“代码视图”对代码的细节进行修改。

注意：由于网页本质上是 HTML 代码，在设计视图中的可视化操作实质上仍然是编写代码。因此可以在设计视图中完成的工作一定也可以在代码视图中完成，也就是说编写代码方式制作网页是万能的，因此要重视对 HTML 代码的学习。

2.1.3 使用 Dreamweaver 新建 HTML 文件

打开 Dreamweaver，在“文件”菜单中选择“新建”命令(快捷键为 Ctrl＋N)，在新建文档对话框中选择“基本页”→HTML 选项，单击“创建”按钮就会出现网页的设计视图。在设计视图中可输入网页内容，然后保存文件(“文件”→“保存”菜单命令，快捷键为 Ctrl＋S，第一次保存时要求输入网页的文件名)，就新建了一个 HTML 文件，最后可以按 F12 键在浏览器中预览网页，也可以在保存的文件夹中找到该文件双击运行。

注意：网页在 Dreamweaver 设计视图中的效果和浏览器中显示的效果并不完全相同，所以测试网页时应按 F12 键在浏览器中预览最终效果。

2.1.4 HTML 标记

标记(tags，也称标签)是 HTML 文档中一些有特定意义的符号，这些符号用来指明内容的含义或结构。HTML 标记由一对尖括号“＜＞”和标记名组成。标记分为“起始标记”和“结束标记”两种。两者的标记名是相同的，只是结束标记名前多了一个“/”。例如在图 2-4 中，＜b＞为起始标记，＜/b＞为结束标记，其中“b”是标记名称，表示内容为粗体。HTML 标记名是大小写不敏感的。例如，＜b＞…＜/b＞和＜B＞…＜/B＞的效果都是一样的，但是 XHTML 标准规定，标记名必须小写，因此应注意用小写字母书写。

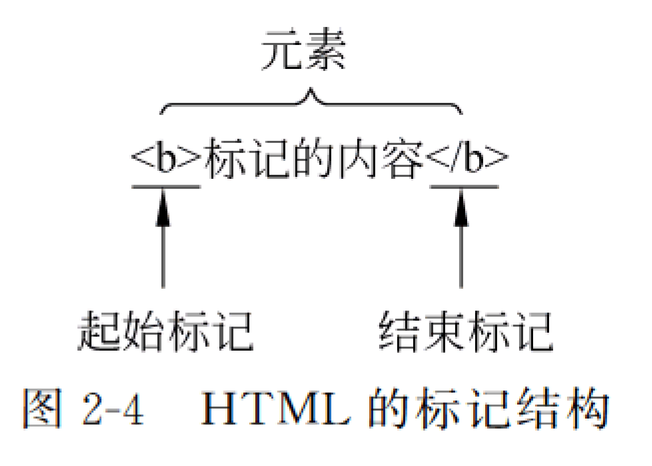


图 2-4 HTML 的标记结构

1. 单标记和双标记

大多数标记都是成对出现的，称为双标记，如＜p＞…＜/p＞、＜table＞…＜/table＞。有少数标记只有起始标记，这样的标记称为单标记，如换行标记＜br /＞，其中 br 是标记名，它是英文“break row”(换行)的缩写。XHTML 规定单标记也必须封闭，因此在单标记名后应以斜杠结束。

2. 标记带有属性时的结构

实际上，标记一般还可以带有若干属性(attribute)，属性用来对元素的特征进行具体描述。属性只能放在起始标记中，属性和属性之间用空格隔开，属性包括属性名和属性值(value)，它们之间用“＝”分开，如图 2-5 所示。

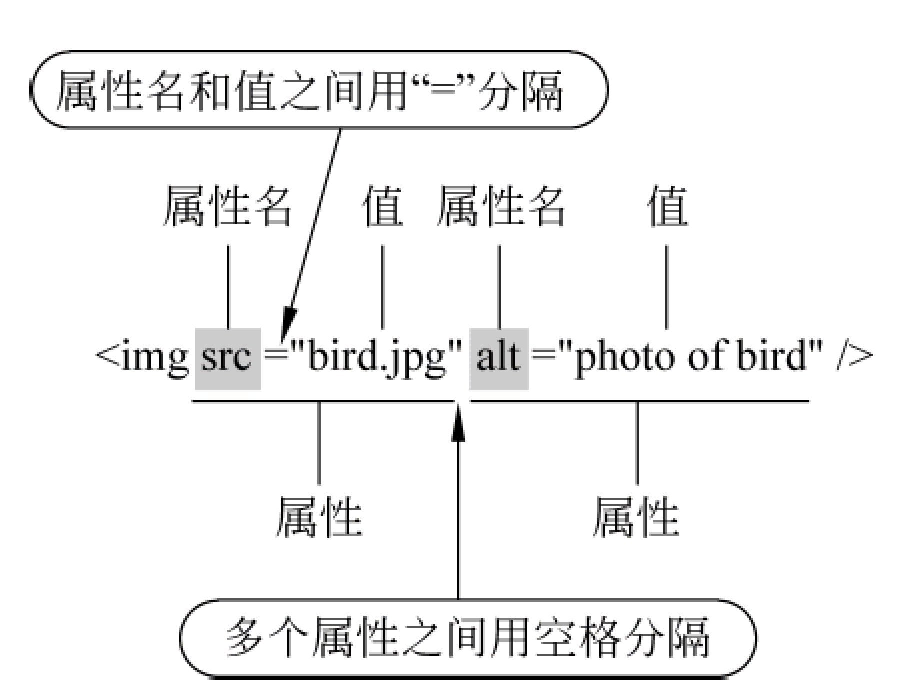


图 2-5　带有属性的 HTML 标记结构

例 2.1　讨论下列 HTML 标记的写法错在什么地方？(答案略)

(1) ＜img "birthday.jpg " /＞

(2) ＜i＞ Congratulations! ＜i＞

(3) ＜a href＝"file.html"＞linked text＜/a href＝"file.html"＞

(4) ＜p＞This is a new paragraph＜\p＞

(5) ＜li＞The list item＜/li＞

提示：把 HTML 标记(如＜p＞、＜/p＞)和标记之间的内容的组合称为 HTML 元素。HTML 元素可分为“有内容的元素”和“空元素”两种。“有内容的元素”是由起始标记、结束标记和两者之间的内容组成的，其中元素内容既可以是文字内容，也可以是其他元素。“空元素”是只有起始标记而没有结束标记和元素内容的元素，如＜br /＞。

2.1.5　常见的 HTML 标记及属性

网页中的文本、图像、超链接、表格等各种元素，其实质上都是使用对应的 HTML 标记实现的。要在网页中添加各种网页元素，只要在 HTML 代码中插入对应的 HTML 标记并设置属性即可。HTML 5 中定义的标记总共有 100 多个，但是常用的 HTML 标记只有下面列出的 40 多个，这些标记及其含义必须熟记下来。表 2-1 对标记按用途进行了分类。

表 2-1　HTML 标记的分类

类　　别	标 记 名 称
文档结构	html，head，body
头部标记	title，meta，link，style，script，base
文本结构标记	p，h1～h6，pre，marquee，br，hr
字体标记	font，b，i，u，strong，em
列表标记	ul，ol，li，dl，dt，dd
超链接标记	a，map，area
图像及媒体元素标记	img，embed，object
表格标记	table，tr，td，th，tbody
表单标记	form，input，textarea，select，option，fieldset，legend，label
容器标记	div，span

HTML 语言还为标记定义了许多的属性，有些属性是可以用在任何标记中的，称为公共属性；而大部分属性是某些标记独有的，称为特有属性。表 2-2 列出了所有 HTML 标记具有的公共属性和某些标记的特有属性。

表 2-2 HTML 标记的一些常见的属性

公共属性	含　　义	特有属性	含　　义
style	为元素引入行内 CSS 样式	align	定义元素的水平对齐方式
class	为元素定义一个类名	src	定义元素引用的文件的 URL
id	为元素定义一个唯一的 id 名	href	定义超链接所指向的文件的 URL
name	为元素定义一个名字	target	定义超链接中目标文件的打开方式
title	定义鼠标悬停在元素上时的提示文字	border	设置元素的边框宽度

2.2 在网页中添加文本和图像

在网页中，文本和图像是最基本的两种网页元素，文本和图像在网页中可以起到传递信息、美化页面、点明主题等作用。在网页中添加文本和图像并不难，主要问题是如何编排这些内容以及控制它们的显示方式，让文本和图像看上去编排有序，整齐美观。

要在网页中添加文本、图像等各种网页元素，只要在 HTML 代码中插入对应的 HTML 标记并设置属性和内容即可。

2.2.1 创建文本和列表

在网页中添加文本的方式主要有以下几种：

1. 直接写文本

这是最简单的插入文本方法，有时候文本并不需要放在文本标记中，完全可直接放在其他标记中。例如，<div>文本</div>、<td>文本</td>、<body>文本</body>、
文本。

2. 用段落标记<p>…</p>格式化文本

各段落文本将换行显示，段落与段落之间有一行的间距。例如：

```
<p>第一段</p><p>第二段</p><p>第三段</p>
```

3. 用标题标记<hn>…</hn>格式化文本

标题标记是具有语义的标记，它指明标记内的内容是一个标题。标题标记共有 6 种，用来定义第 n 级标题（n=1～6），n 的值越大，字越小，所以<h1>是最大的标题标记，而<h6>是最小的标题标记。标题标记中的文本将以粗体显示，实际上可看成是特殊的段落标记。

标题标记和段落标记均具有对齐属性 align，用来设置元素的内容在元素占据的一行空间内的对齐方式。该属性的取值有 left（左对齐）、right（右对齐）、center（居中对齐）。

4. 文本换行标记

标记是强制换行标记，如果希望 HTML 代码中的文本在浏览器中换行，可在要换行处插入
标记。在 Dreamweaver 中插入
标记的快捷键是 Shift+Enter。

5. 列表标记

为了合理地组织文本或其他元素，网页中经常要用到列表。列表标记分为无序列表<ul>、有序列表<ol>和定义列表<dl>三种。每个列表标记都是配对标记，在列表标记中可包含若干个<li>标记，表示列表项。

图 2-6 是一个包含了各种文本和列表的网页，其对应的 HTML 代码如下：

```
<html><body>
    <h2 align="center">网页制作语言</h2>
    <p>Web开发领域常用的网页制作语言如下:</p>
  <ul>
      <li>HTML: 网页结构语言</li>
      <li>CSS: 网页表现语言</li>
      <li>JavaScript<br>一种浏览器编程语言</li>
      <li>PHP</li>
  </ul>
</body></html>
```

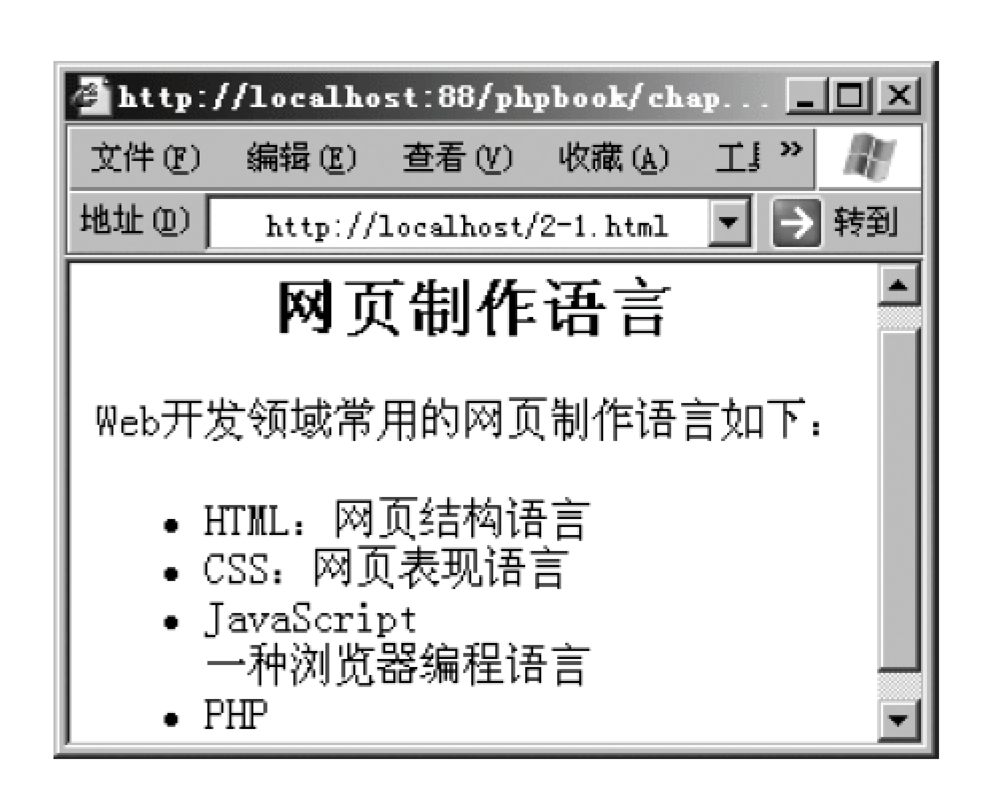


图 2-6　包含各种文本标记的网页

提示：换行标记
不会产生空行，而两个段落标记<p>之间会有一行的空隙。

2.2.2　插入图像

网页中图像对浏览器者的吸引力远远大于文本，选择最恰当的图像，能够牢牢吸引浏

览者的视线。图像直接表现主题,并且凭借图像的意境,使浏览者产生共鸣。缺少图像而只有色彩和文字的设计,给人的印象是没有主题的空虚的画面,浏览者将很难了解该网页的主要内容。

1. 使用<img>标记插入图像文件

在 HTML 中,用<img>标记可以插入图像文件,并可设置图像的大小、对齐等属性,它是一个单标记,如图 2-7 所示的网页中插入了一张图片,其对应的 HTML 代码如下:

```
<html><body>
<p>今天钓到一条大鱼,好高兴!</p>
<img src="images/dayu.jpg" width="200"
height="132" align="center"
title="好大的鱼"/>
</body></html>
```



图 2-7　在网页中插入图片

该网页中显示的图片文件位于当前文件所在目录下的 images 目录中,文件名为 dayu.jpg,如果不存在该文件,则会显示一片空白。<img>标记的常见属性如表 2-3 所示。

表 2-3　<img>标记的常见属性

属　　性	含　　义
src	图片文件的 URL 地址
alt	当图片无法显示时显示的替代文字
title	光标停留在图片上时显示的说明文字
align	图片的对齐方式,共有 9 种取值
width、height	图片在网页中的宽和高,单位为像素或百分比

在 Dreamweaver 中,单击工具栏中的图像按钮( )可让用户选择插入一张图片,其实质是 Dreamweaver 在代码中自动插入了一个<img>标记,选中插入的图像,还可在属性面板中设置图像的各种属性以及图像的链接地址等。

除了使用<img>标记插入图像外,还可将图像作为 HTML 元素的背景嵌入网页中,由于 CSS 的背景属性功能强大,现在更推荐将元素的装饰性图像都作为背景嵌入。如果图像是通过<img>元素插入的,则可以在浏览器上通过按住鼠标左键拖动选中图片,选中后图片呈现反选状态,还可以将它拖动到地址栏里,那么浏览器将单独打开这幅图片。如果是作为背景嵌入,则无法选中图片。

2. 网页中支持的图像文件格式

网页中可以插入的图像文件格式有 JPG、GIF 和 PNG 格式,它们都是压缩形式的图像格式,容量较位图格式(BMP)的图像小,适合于网络传输。这三种格式图像文件的特

点如表 2-4 所示。

表 2-4　网页中 3 种图像格式的比较

图像格式	JPG	GIF	PNG
压缩形式	有损压缩	无损压缩	无损压缩
支持的颜色数	24 位真彩色	256 色	真彩色或 256 色
支持透明	不支持	支持全透明	支持半透明和全透明
支持动画	不支持	支持	不支持
适用场合	照片等颜色丰富的图片	卡通图形、线条、图标等颜色数少的图片	都可以

2.3　利用 Dreamweaver 代码视图提高效率

Dreamweaver 提供了方便的代码编写功能。前面曾谈到，页面在浏览器中的最终显示效果完全是由 HTML 代码决定的，Dreamweaver 只是帮助用户方便地插入或者生成必要的代码。在实际工作中，还是会经常遇到通过可视化的方式生成的代码并不能满足需要的情况，这时就需要设计师对代码进行手工调整，这个工作可以在 Dreamweaver 的代码视图中完成。

在代码视图中，Dreamweaver 提供了很多方便的功能，可以帮助用户更高效地完成代码的输入和编辑操作。

2.3.1　代码提示

在 HTML 和 CSS 语言中，都有很多标记、属性和属性值，设计者要把繁多的标记、属性和属性值记清楚是很不容易的。为此，Dreamweaver 提供了方便的代码提示功能，以减少设计者的记忆量，并加快代码的输入速度。

在 Dreamweaver 的“代码”视图中，如果希望在代码中的某个位置增加一个 HTML 标记，只需把光标移动到目标位置，输入“＜”，就会弹出标记提示下拉框，如图 2-8 所示。这时可以按↓键选取所需的标记，再按回车键即完成对该标记的输入，有效地避免了拼写错误。

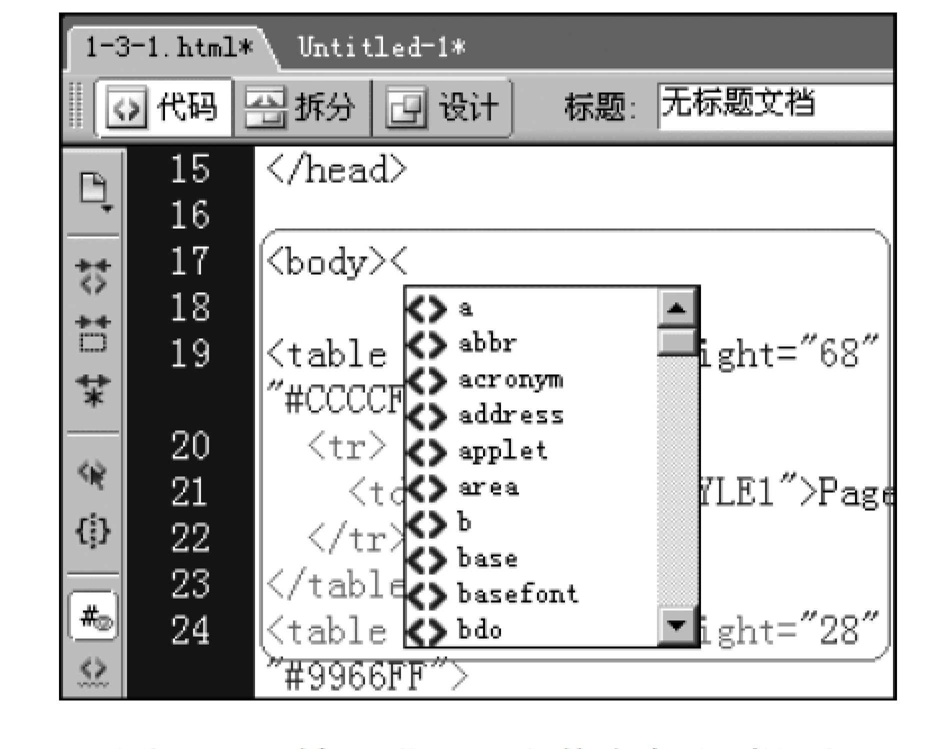


图 2-8　输入“＜”后弹出标记提示

如果要为标记添加一个属性，只需在标记名或其属性后按下空格键，就会出现下拉框，列出了该标记具有的所有属性和事件，如图 2-9 所示，按↓键就可选取所需的属性。实际上，通过查看列出的所有属性，还可以帮助我们学习该标记具有哪些属性。

如果列出的属性特别多，那么可以继续输入所需属性的第一个字母，这时属性提示框

中的内容会发生变化，仅列出以这个字母开头的属性，就大大缩小了选择范围。

在选择了某个属性后，按回车键，Dreamweaver 的代码提示功能就会自动输入“=""”，并会弹出备选的属性值，如图 2-10 所示。这时按↓键就可选取属性值，再按回车键即完成了属性值的输入。如果要修改属性值，只需把属性值连同引号一起删掉，然后再输入一个双引号，就会再次弹出属性值提示框了。

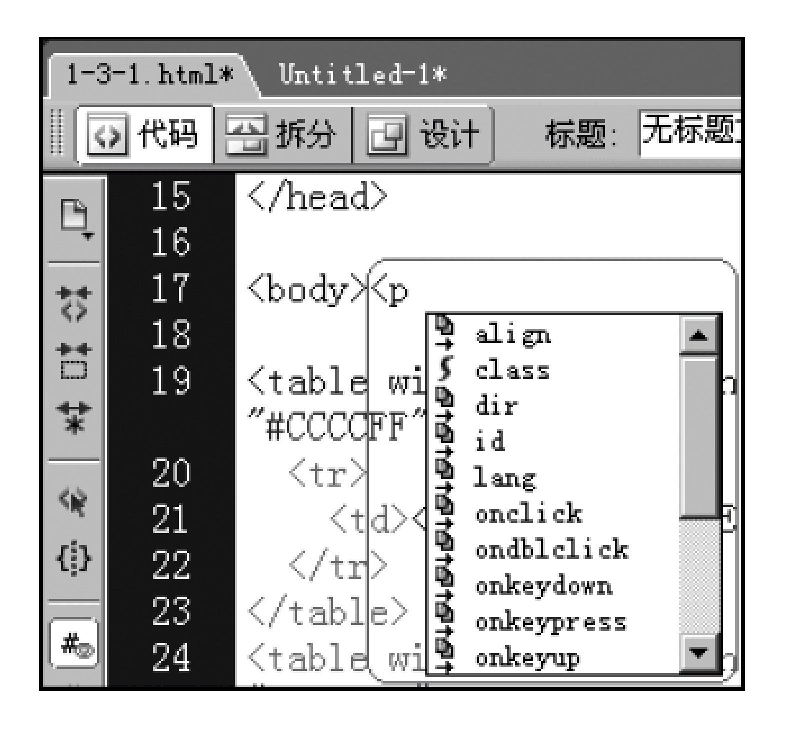


图 2-9　输入空格后弹出属性提示



图 2-10　选中属性后弹出属性值提示

2.3.2　Dreamweaver 中的常用快捷键

表 2-5 列出了 Dreamweaver 的一些常用快捷键，实际上这些快捷键是很多软件通用的快捷键，在其他很多应用软件(如 Word、Fireworks)中也经常使用。

表 2-5　Dreamweaver 的常用快捷键

快捷键	功　能	快捷键	功　能
Ctrl+Z	撤销操作	Ctrl+C	复制
Ctrl+S	保存文档	Ctrl+V	粘贴
F12	预览网页	Ctrl+X	剪切
Ctrl+A	全选	Ctrl+N	新建文档

1. Ctrl+Z

在制作网页过程中，为了调试网页，经常会把网页修改得很乱，此时如果想回退到原来的状态，只需按 Ctrl+Z 键进行撤销操作，连续按则能撤销多步操作。因此 Ctrl+Z 键可能是网页制作中使用最多的快捷键。需要注意的是，即使文档保存过，但如果没有将文档的窗口关闭就仍然能按 Ctrl+Z 键进行撤销。

2. Ctrl+S

由于调试网页时经常需要预览网页，而预览之前必须先保存网页，因此 Ctrl+S 键也是用得很频繁的快捷键，它的作用是保存网页，调试过程中通常是按了 Ctrl+S 键后马上按 F12 键预览。

3. Ctrl+A、Ctrl+C、Ctrl+V、Ctrl+X

这几个快捷键是文本编辑中最常用的快捷键，在制作网页过程中经常需要使用。例如在网上找到一个完整的 HTML 源代码，想在 Dreamweaver 中调试。那么最快捷的方式就是先在网上复制这段代码，然后在 Dreamweaver 中按 Ctrl+N 键新建网页，切换到代码视图，按 Ctrl+A 键全选代码视图中的代码，再按 Ctrl+V 键粘贴就能将网上的代码替换掉 Dreamweaver 中原来的代码。

2.4 创建超链接

超链接是组成网站的基本元素，通过超链接可以将很多网页链接成一个网站，并将 Internet 上的各个网站联系在一起，浏览者可以方便地从一个网页跳转到另一个网页。

超链接是通过 URL(统一资源定位器)来定位目标信息的。URL 主要包括 4 部分：网络协议(如 http://)、域名或 IP 地址、文件路径、文件名。

2.4.1 超链接标记<a>

在网页中，<a>…标记且带有 href 属性时表示是超链接，如图 2-11 所示的网页中创建了 2 个超链接，当鼠标移动到超链接上时会变成手形。其代码如下：

```
<html><body>
  <a href="/index.html" target="_blank">网站首页</a>
  <a href="mailto:xia@qq.com" title="欢迎给我来信">联系我们</a>
</body></html>
```

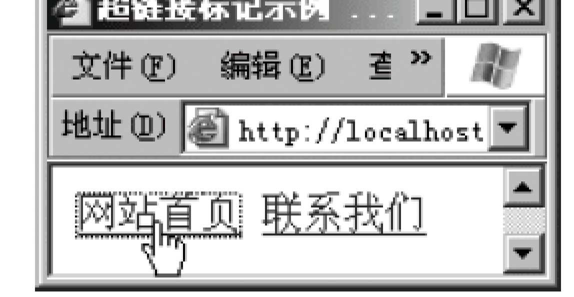


图 2-11 网页中的超链接

<a>标记的属性及其取值如表 2-6 所示。

表 2-6 <a>标记的属性及其取值

属性名	说　明	属　性　值
href	超链接的 URL 路径	相对路径或绝对路径、Email、# 锚点名
target	超链接的打开方式	_blank：在新窗口打开 _self：在当前窗口打开，默认值 _parent：在当前窗口的父窗口打开 _top：在整个浏览器窗口打开链接 窗口或框架名：在指定名称的窗口或框架中打开
title	超链接上的提示文字	属性值是任何字符串
id、name	锚点的 id 或名称	自定义的名称，如 id="ch1"。<a>标记作为锚点使用时，不能设置 href 属性

超链接的源对象是指可以设置链接的网页对象，主要有文本、图像或文本图像的混合

体，它们对应<a>标记的内容，另外还有热区链接。在 Dreamweaver 中，这些网页对象的属性面板中都有“链接”设置项，可以很方便地为它们建立链接。

1. 用文本制作超链接

在 Dreamweaver 中，可以先输入文本，然后用鼠标选中文本，在属性面板的“链接”文本框中输入链接的地址并按 Enter 键；也可以单击“常用”工具栏中的“超级链接”图标，在对话框中输入“文本”和链接地址；还可以在代码视图中直接写代码。无论使用何种方式，生成的超链接代码类似于下面这种形式：

```
<a href="index.htm" target="_blank">首页</a>
```

2. 用图像制作超链接

首先需要插入一幅图片，然后选中图片，在属性面板的“链接”文本框中设置图像链接的地址。生成的代码如下：

```
<a href="index.htm"><img src="images/info.gif" title="返回首页" border="0" /></a>
```

用图像制作超链接，最好设置<img>标记的 border 属性等于 0，否则图像周围会出现一个蓝色的 2 像素粗的边框，很不美观。

3. 热区链接

用图像制作超链接只能让整张图片指向一个链接，那么能否在一张图片上创建多个超链接呢？这时就需要热区链接。所谓热区链接，就是在图片上划出若干个区域，让每个区域分别链接到不同的网页。比如一张中国地图，单击不同的省份会链接到不同的网页，就是通过热区链接实现的。

制作热区链接首先要插入一张图片，然后选中图片，在展开的图像“属性”面板上有“地图”选项，它的下方有三个小按钮分别是绘制矩形、圆形、多边形热区的工具，如图 2-12 所示。可以使用它们在图像上拖动绘制热区，也可以使用箭头按钮调整热区的位置。

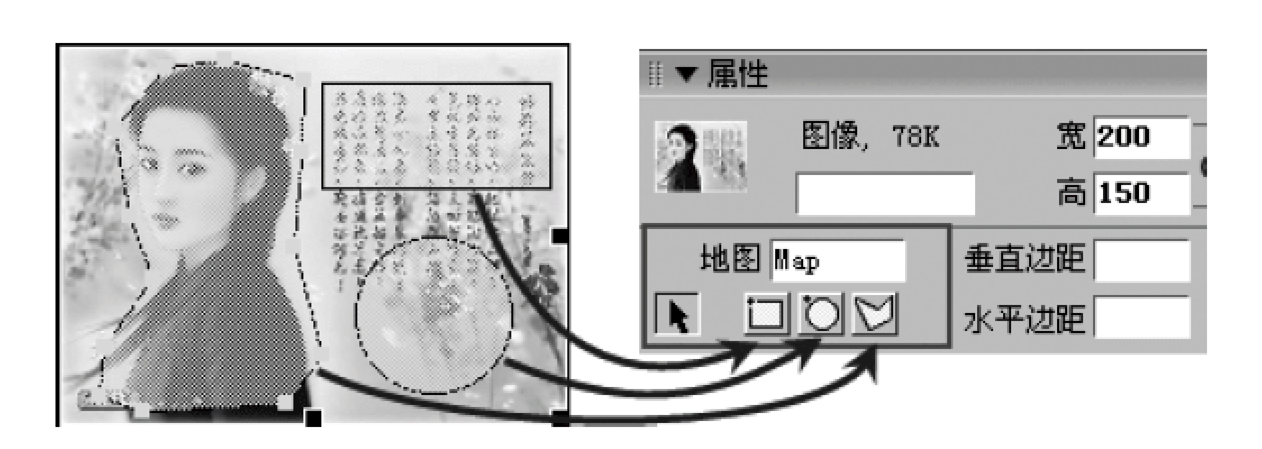


图 2-12　图像属性面板中的地图工具

绘制了热区后，可看到在 HTML 代码中增加了<map>标记，表示在图像上定义了一幅地图。地图就是热区的集合，每个热区用<area>单标记定义，因此<map>和<area>是成组出现的标记对。定义热区后生成的代码如下：

```
<img src="images/xf.jpg" alt="说明文字" border="0" usemap="#Map" />
<map name="Map" id="Map">
```

```
    <area shape="rect" coords="51,131,188,183" href="title.htm" alt="说明文字" />
    <area shape="rect" coords="313,129,450,180" href="#h3" />
</map>
```

其中,<img>标记会增加 usemap 属性与它上面定义的地图(热区)建立关联。

<area>标记的 shape 属性定义了热区的形状,coords 属性定义了热区的坐标点,href 属性定义了热区链接的文件,alt 属性可设置鼠标移动到热区上时显示的提示文字。

2.4.2 绝对 URL 与相对 URL

URL 是统一资源定位器的意思。在网页中,URL 用来描述链接的文件或调用的图片的地址。网页中的 URL 可分为绝对 URL 和相对 URL。

1. 绝对 URL(绝对路径)

绝对 URL 是采用完整的 URL 来规定文件在 Internet 上的精确地点,包括完整的协议类型、计算机域名或 IP 地址、包含路径信息的文档名。书写格式为:协议://计算机域名或 IP 地址[/文档路径][/文档名]。例如:

```
<a href="http://www.hynu.cn/index.htm">学院首页</a>        <!--链接文件-->
<img src=" http://www.hynu.cn/images/bg.jpg" />            <!--调用图片-->
```

2. 相对 URL(相对路径)

相对 URL 是相对于当前页的地点来规定文件的地点。应尽量使用相对 URL 创建链接,使用相对路径创建的链接可根据目标文件与当前文件的目录关系,分为四种情况:

(1) 如果要链接到同一目录内的其他文件,直接写目标文件名即可。

```
<a href="目标文件名">链接文本</a>
```

(2) 链接到下一级目录中的文件,则先写下一级目录名带"/",再写目标文件名。

```
<a href="子目录名/目标文件名">链接文本</a>
```

(3) 如果要链接到上一级目录中的文件,则在目标文件名前添加"../",因为".."表示上级目录,而"."表示本级目录。例如,

```
<a href="../目标文件名">链接文本</a>
```

(4) 链接到上一级目录中其他子目录中的网页文件,则可先退回到上一级目录,再进入目标文件所在的目录,格式为:

```
<a href="../子目录名/目标文件名">链接文本</a>
```

3. 相对 URL 使用举例

下面举个例子说明相对路径的使用方法。网站的文件目录结构如图 2-13 所示。

图中的矩形表示文件夹,圆角矩形表示文件。

图 2-13 网站的文件目录结构

(1) 如果 f-01 目录下的 a. htm 需要显示同目录下的 box. gif 图片,因为在当前目录下可以直接找到 box. gif 文件,所以相对路径是 box. gif。

(2) 如果根目录下的 index. htm 需要显示 f-01 目录下的 box. gif 图片,则应先进入 f-01 目录,再找到 box. gif 文件,因此相对路径是“f-01/box. gif”。

(3) 如果 f-03/02 目录下的 b. htm 需要显示 01 目录下的 cat. gif 图片,则应从 02 目录退一级到 f-03 目录,再进入 01 目录,所以相对路径是“../01/cat. gif”。

(4) 如果 b. htm 需要显示 box. gif 图片,应该写成“../../f-01/box. gif”。

(5) 如果 a. htm 需要显示 cup. gif 图片,应该写成“../f-03/02/cup. gif”。

可见,相对路径比较简便,不需输入完整的 URL。另外相对路径还有一个很明显的优点是:可以毫无顾忌地修改网站的域名或网站文件夹在服务器硬盘中的存放位置。

提示:如果在 Dreamweaver 中制作网页时看到代码中 URL 为 file 协议的格式,例如:file:///E|/网页制作上课/DEMO/bg. png,说明网页中引用的资源是本机上的,出现这种情况的原因是引用的文件没在网站目录内,或根本没创建网站目录,或网页文件尚未保存到网站目录内。当网页上传到服务器后,由于该资源在服务器上的存放路径和本机上的路径一般不会相同,就会出现找不到文件的情况,因此应避免这种情况出现。

2.4.3 超链接的种类(href 属性的取值)

超链接有很多种类,如网页链接、电子邮件链接、锚链接等,这些不同类的超链接区别在于其 href 属性的取值不同。因此可以根据 href 属性的取值来划分超链接的类型。

1. 链接到其他网页或文件

因为超链接本身就是为了把 Internet 上各种网页或文件链接在一起,所以链接到文件的链接是最重要的一类超链接,它可分为以下几种:

- 内部链接,链接地址是相对 URL,如

```
<a href="../index.htm">首页</a>
```

- 外部链接,链接地址是绝对 URL,如

```
<a href="http://www.qq.com">腾讯网</a>
```

- 下载链接,链接地址是一个浏览器不能打开的文件类型,如 rar、doc、apk 等,单击链接会弹出文件下载框,如

```
<a href="inc/test.rar">点击下载</a>
```

2. 电子邮件链接

如果在链接的URL地址前面有“mailto:”，就表示是电子邮件链接，单击电子邮件链接后，浏览器会自动打开默认的电子邮件客户端程序(如Outlook)。

```
<a href="mailto:xiaoli@163.com">xiaoli@163.com</a>
```

由于我国用户大多不喜欢使用客户端程序发送邮件，所以也可以不建立电子邮件链接，直接把Email地址作为文本写在网页上，这样还可以防止垃圾邮件的侵扰。

3. 锚链接(链接到页面中某一指定的位置)

当网页内容很长，需要进行页内跳转链接时，就需要定义锚点和锚点链接，锚点可使用name属性或id属性定义。锚链接需要和锚点配合使用，单击锚链接会跳转到指定的锚点处。示例代码如下：

```
<a id="ch4"></a>                <!--定义锚点,锚点名为 ch4 -->
<a href="#ch4">…</a>           <!--链接到当前网页的锚点 ch4 处 -->
<a href="intro.htm#ch4">…</a>   <!--链接到 intro.htm 网页的锚点 ch4 处 -->
```

注意：定义锚点时锚点名前面不要加#号，链接到锚点时锚点名前要加#号。

4. 空链接和脚本链接

还有一些有特殊用途的链接，例如测试网页时用的空链接和脚本链接等。

```
<a href="#">…</a>              <!--空链接,网页会返回页面顶端-->
<a href="JavaScript:self.close();">关闭窗口</a>    <!--脚本链接-->
```

5. HTML 5中新增的超链接

在HTML 5中，为了方便制作手机网站，超链接的href属性有了更多的取值，例如：

- 拨打电话号码的链接。

```
<a href="tel:13335418888">拨打电话</a>
```

- 发短信的链接。

```
<a href="sms:18688888888">发短信</a>
```

2.4.4 超链接目标的打开方式

超链接标记<a>具有target属性，用于设置链接目标的打开方式。在Dreamweaver中在“目标”下拉列表框中可设置target的属性的取值，如图2-14所示。其常用的取值有四种。



图2-14 “目标”下拉列表框

(1) _self：在原来的窗口或框架打开链接的网页，这是target

属性的默认值，因此也可以不写。

（2）_blank：在一个新窗口打开所链接的网页，这个很有用，可防止打开新网页后把原来的网页覆盖掉，例如：

```
<a href="http://www.rongshu.com" target="_blank">榕树下</a>
```

（3）_parent：将链接的文件载入到父框架打开，如果包含的链接不是嵌套框架，则所链接的文档将载入到整个浏览器窗口。

（4）_top：在整个浏览器窗口载入所链接的文档，因而会删除所有框架。

在这四种取值中，“_parent”“_top”仅仅在网页被嵌入到其他网页中有效，如框架中的网页，所以它们用得很少。用得最多的还是通过 target 属性使网页在新窗口中打开，如 target＝"_blank "。要注意不要漏写取值名称前的下划线“_”。

2.4.5 超链接制作的原则

1. 可以使用相对链接尽量不要使用绝对链接

相对链接的好处在前面已经详细介绍过，原则上，同一网站内文件之间的链接都应使用相对链接方式，只有在链接到其他网站的资源时才使用绝对链接。例如，和首页在同一级目录下的其他网页要链接到首页，有如下三种方法：

（1）<a href＝". ">首 页 <!--链接到本级目录，则自动打开本级目录的主页-->

（2）<a href＝"index. html">首 页 <!--链接到首页文件名-->

（3）<a href＝"http://www. hynu. cn">首 页 <!--链接到网站名-->

通常应该尽量采用前两种方法，而不要采用第三种方法。但第一种方式需要在 Web 服务器上设置网站的首页为 index. html 后才能正确链接，这给在文件夹中预览网页带来了不便。

2. 链接目标尽可能简单

假如要链接到其他网站的主页，那么有如下两种写法：

（1）<a href＝"http://www. hynu. cn">首 页

（2）<a href＝"http://www. hynu. cn/index. html">首 页

则第（1）种写法比第（2）种写法要好，因为第（1）种写法不仅简单，还可以防止以后该网站将首页改名（如将 index. html 改成 index. jsp）造成链接不上的问题。

3. 超链接的综合运用实例

下面这段代码包含了各种类型的超链接，请认真总结这些超链接的写法。

```
<html><body>
<p><a href="dance.html">红舞鞋</a></p>
<p><a href="#xrh">雪绒花</a></p>
<p><a href=mailto:xiali@163.net title="欢迎给我来信"><img src="mail.gif" /></a></p>
```

```
<p>好站推荐:<a href="http://www.baidu.com" target="_blank">百度</a></p>
<p><a id="xrh"></a>雪绒花的介绍……</p>
<p align="right"><a href="JavaScript:self.close();">关闭窗口</a></p>
</body></html>
```

2.4.6 Dreamweaver 中超链接属性面板的使用

Dreamweaver 中建立链接的选项框如图 2-15 所示，文字、链接、图像和热区的属性面板中都有“链接”这一项。其中，“链接”对应标记的 href 属性，“目标”对应 target 属性。利用超链接属性面板可快速建立超链接，首先选中要建立超链接的文字或图片，然后在“链接”选项框中输入要链接的 URL 地址。



图 2-15 Dreamweaver 中的建立链接选项框

其中在链接地址栏输入 URL 有三种方法：一是直接在文本框输入 URL；二是单击“文件夹”图标浏览找到要链接的文件；三是按住拖放定位图标( )不放将其拖动到锚点处或文件面板中要链接的文件上，如图 2-16 所示。使用以上任何一种方式使“链接”框中出现了内容后，“目标”下拉列表框将变为可用，可选择超链接的打开方式。

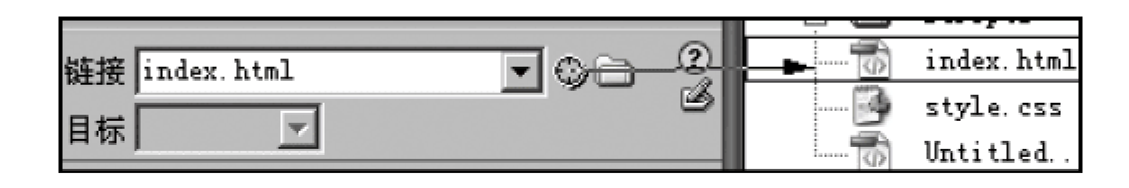


图 2-16 使用拖动链接定位图标方式建立链接

2.5 插入 Flash 及多媒体元素

Flash 是网络上传输的矢量动画，利用 Dreamweaver 可以很方便地在网页中插入 Flash 文件，从而在网页上展现丰富的动画效果，而网页中插入视频的方法和插入 Flash 的方法差不多，也是通过插件或 ActiveX 方式插入的。

2.5.1 插入 Flash

1. 使用 Dreamweaver 在网页中插入 Flash 的两种方法

(1) 执行“插入”→“媒体”→Flash 菜单命令，再在“属性”面板中调节插件的宽和高，在代码视图中可看到插入 Flash 元素是通过同时插入<object>标记和<embed>标记实现的，以确保在所有浏览器中都获得应有的效果。

(2) 执行“插入”→“媒体”→“插件”菜单命令，此方法在代码中仅插入了<embed>标记。如果不需要设置特别的参数(如 wmode="transparent")，那么在 IE 和 Firefox 也能看到效果，而且代码更简洁，所以推荐用这种方式。

2. 在图像上放置透明 Flash

有些 Flash 动画的背景是透明的，在百度上搜索“透明 Flash”可以找到很多透明的 Flash 动画。可以将这种透明背景的 Flash 动画放在一幅图片上，使图片看起来和 Flash 融为一体也有动画效果了。方法是先将一张需要放置透明 Flash 的图片作为单元格（或 div 等其他元素）的背景，然后在此单元格内插入一个透明 Flash 文件，这样这个 Flash 文件就覆盖在了图片的上方。然后调整此 Flash 插件的大小与图片大小相一致。再选中该 Flash 插件，单击属性面板里的“参数”按钮，在如图 2-17 所示的“参数”对话框中新建一个参数 wmode，值设置为 transparent。生成的代码包括一个<param>标记和一个在<object>标记中的 wmode 属性，其中<param name="wmode" value="transparent"/>使 Flash 在 IE 中能够透明，而 wmode="transparent"使 Flash 在 Firefox 中透明。

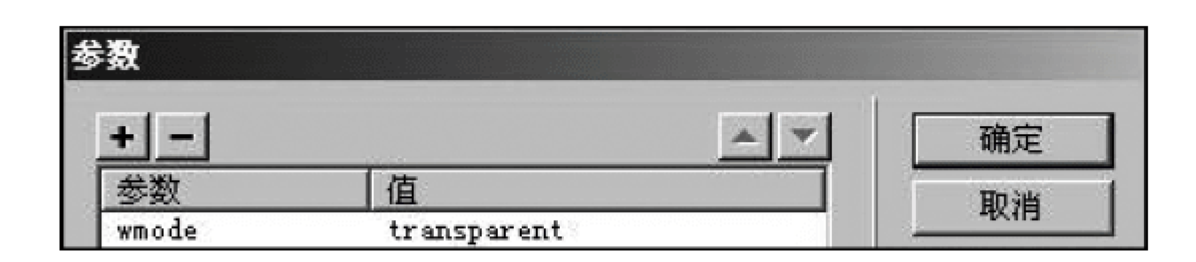


图 2-17　设置 Flash 文件透明方式显示

2.5.2　插入视频或音频文件

1. 插入 avi、mpg 或 wmv 格式视频

视频文件的格式主要有 avi、wmv、mpg、rm、rmvb 等，如果要插入 avi、mpg 或 wmv 等 Windows 媒体播放器能播放的格式文件，可以直接使用插件方式插入。方法是在 Dreamweaver 中执行“插入”→“媒体”→“插件”菜单命令，然后在插件的“属性”面板中设置视频文件的宽和高，如图 2-18 所示。注意高度的设置值最好比视频的高度多 40 像素，因为这个高度包含了播放控制条的高度。这样在网页中就可以播放视频了。

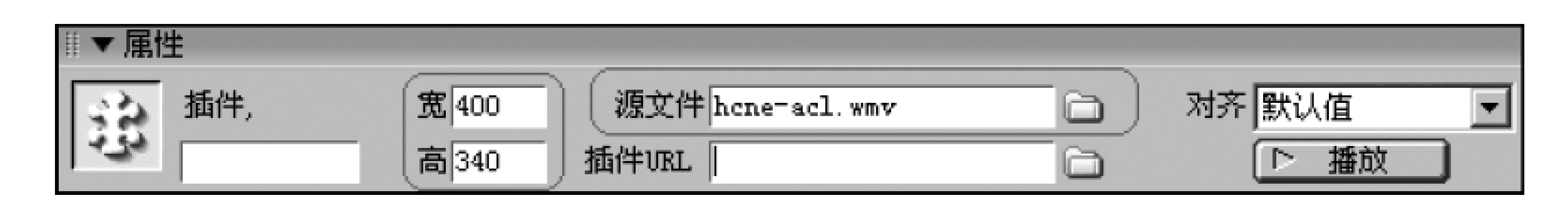


图 2-18　插件属性面板

切换到代码视图，可看到生成的源代码如下：

```
<embed src="acl.wmv" width="400" height="340"></embed>
```

如果不希望在打开网页后视频自动播放，可添加属性 autostart="false"。

2. 插入 RealPlayer 格式视频

要在网页中插入 rm、rmvb 等 RealPlayer 格式的视频也可使用<embed>标记来插入，但必须将网页上传到服务器上才能播放。为了在本机上就能播放，可以使用 ActiveX 方式插入。方法是执行“插入”→“媒体”→ActiveX 菜单命令，这样就在网页中插入了一

个 ActiveX 控件，然后再设置它的宽和高，对于 RealPlayer 格式的视频，要在“属性”面板中将 ClassID 设置为“RealPlayer/…”，如图 2-19 所示。



图 2-19　ActiveX 属性面板

然后再在代码视图对<object>标记中设置参数<param>，主要是指定视频文件的 URL 和关联播放组件。源代码如下：

```
<object classid="clsid:CFCDAA03-8BE4-11cf-B84B-0020AFBBCCFA" width="452" height="320">
    <param name="console" value="clip1" />    <!--用来关联播放组件-->
    <param name="autostart" value="true" />
    <param name="src" value="wenrou.rm" />    <!--指定文件的 URL-->
    <param name="controls" value="imagewindow" />
</object>
```

3. 插入 FLV 格式视频

FLV(Flash Video)格式的视频是目前使用最广泛的网络流媒体视频传播格式，被许多在线视频网站采用。在网页中插入 FLV 视频的方法有两种。

(1) 执行菜单命令“插入”→“媒体”→“Flash 视频”，在弹出的“插入 Flash 视频”对话框中，选择视频文件的 URL，在“外观”中可以选择一种播放器的外观，最后要选中“自动播放”复选框。这样就能在网页中播放 FLV 视频了。

(2) 使用 swfobject.js 文件，也能插入 FLV 视频。该文件需要配合 mediaplayer.swf 文件使用，将上述两个文件放在网站目录下，并在网页中插入如下 JavaScript 代码，就能播放 FLV 视频了。

```
<script src="swfobject.js"></script>
<script>
    var s1=new SWFObject("mediaplayer.swf","mediaplayer","640","480","7");
    s1.addParam("allowfullscreen","true");
    s1.addVariable("width","640");          //设置视频的宽度
    s1.addVariable("height","480");
    s1.addVariable("file","hnfh.flv");      //视频文件的 URL
    s1.addVariable("image","hnfh.jpg");     //视频播放时的初始图像
    s1.write("container");
</script>
```

使用这种方法，能够避免在 HTML 代码中出现<object>和<embed>等非标准标记，从而更加符合 Web 标准，也符合搜索引擎优化的原则。

4. 插入音频文件

插入音频文件同样可用插件方式或 ActiveX 方式实现，下面是插件方式插入的代码：

```
<embed src="wenrou.mp3" width="31" height="26" autostart="false"></embed>
```

如果希望将音频文件设置为背景音乐，即不显示播放器的界面，可加一条隐藏(hidden)属性和循环播放(loop="true")属性。

2.5.3 HTML 5 新增的多媒体标记

在 HTML 4 中，<embed>标记或<object>标记虽然可以插入视频，但支持的视频格式非常有限，这已不能满足网络上播放各种视频文件的需要了。在 HTML 5 中，新增了几个标记用来插入视频或音频文件。其中，<video>标记用于插入视频，<audio>标记用于插入音频，<canvas>标记可以插入生成的图像或动画。

1. <video>标记

<video>标记插入视频的示例代码如下：

```
<video src="movie.ogg" width="320" height="240" controls="controls">
    你的浏览器不支持 video 标记
</video>
```

其中，src 属性用于指定视频文件的路径和文件名，width 和 height 属性设置视频的显示大小，controls 属性用于设置是否显示控制条。

<video>与</video>之间的内容用于在不支持该标记的浏览器中显示替代信息。

由于不同的客户端支持的视频格式有可能不同，为此，<video>标记还提供了设置多个备选视频文件的功能，此时，应使用<source>标记而非 src 属性来设置视频文件的地址。

例如，有些浏览器支持使用了 H.264 编码的 mp4 格式视频文件，但有的浏览器不支持，对于不支持的浏览器，只需要把后备内容放在第 2 个 source 标记中，后备内容可以包含多种视频格式。如果连 video 标记都不支持，还可在其中嵌入 object 标记。下面是示例代码：

```
<video>
    <source src="movie.mp4">
    <source src="movie.ogv">
    <object data="movie.swf"><a href="movie.mp4">download</a>
    </object>
</video>
```

上述代码中，如果浏览器支持 video 标记，也支持 H.264，则会显示第 1 个视频。如果浏览器支持 video 标记，也支持 Ogg，那么会显示第 2 个视频。如果浏览器不支持 video 标记，则会显示 object 标记中的 Flash 视频了。如果浏览器不支持 video 标记，也不

支持 Flash，则会显示下载视频的链接。可见，通过 video 标记，就能为各种浏览器提供支持的视频格式了。

2. <audio>标记

<audio>标记用于插入音频，其用法与 video 标记类似，示例代码如下：

```
<audio src="song.ogg" controls="controls">
    你的浏览器不支持 audio 标记
</audio>
```

<audio>标记也提供了设置多个备选音频文件的功能，此时，应使用 source 标记而非 src 属性来设置音频文件的地址。示例代码如下：

```
<audio controls="controls">
    <source src="song.ogg" type="audio/ogg">
    <source src="song.mp3" type="audio/mpeg">
    你的浏览器不支持 audio 标记
</audio>
```

<audio>标记和<video>标记都具有以下几个属性：

(1) autoplay——如果出现该属性，则媒体文件在网页打开后会自动播放；

(2) controls——如果出现该属性，则播放界面下会显示控制条；

(3) loop——如果出现该属性，则媒体文件会循环播放；

(4) preload——如果出现该属性，则媒体文件在页面加载时就会加载，并预备播放。如果使用了 autoplay 属性，则忽略该属性。

3. <canvas>标记

<canvas>标记称为画布标记，用于在网页上绘制图形。<canvas>标记本身没有绘制图形的能力，所有的绘制工作必须使用 JavaScript 程序来完成。画布是一个矩形区域，我们可以控制其每一像素。<canvas>标记使用的步骤如下：

(1) 创建 canvas 元素，并定义元素的 ID，设置元素的宽度和高度：

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

(2) 通过 JavaScript 获取 canvas 元素，并绘制图形，运行效果如图 2-20 所示。

```
<script>
var c=document.getElementById("myCanvas");        //获取 myCanvas 元素
var cxt=c.getContext("2d");
cxt.fillStyle="#ffff00";                          //设置填充颜色
cxt.fillRect(0,0,150,75);                         //绘制矩形
cxt.moveTo(10,10);                                //将画笔移动到坐标位置
cxt.lineTo(150,50);                               //产生线条
cxt.lineTo(10,50);
```

```
cxt.stroke();                                   //绘制路径
</script>
```

其中，getContext()方法返回一个用于在画布上绘图的环境。该方法的参数目前只能是 2D，它指定是进行二维绘图（目前 canvas 标记不支持 3D 绘图），该方法返回一个环境对象，该对象导出一个二维绘图的 API。

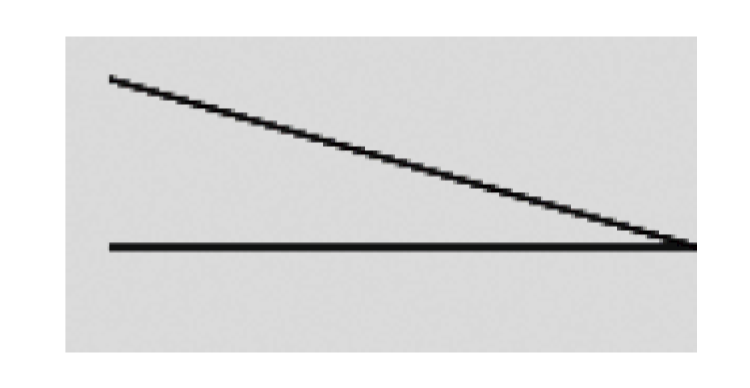


图 2-20　canvas 元素绘制图形的效果

<canvas>标记还可以把一个图像文件放置到画布上，示例代码如下：

```
<canvas id="myCanvas" width="600" height="500">
<script>
var c=document.getElementById("myCanvas");
var cxt=c.getContext("2d");
var img=new Image()
img.src="images/car.jpg"                    //指定图像文件的 URL
cxt.drawImage(img, 10,10,540,460);          //从坐标点 10,10 开始装载图片
</script>
```

其中，drawImage 方法用于在画布上定位图像，并规定图像的宽度和高度。

2.6　创建表格

表格在网页中的应用非常广泛，网页中的表格不仅用来显示数据，还可用来对网页进行排版和布局，使用表格最明显的好处就是能以行列对齐的方式来显示文本或图像信息，以达到精确控制文本和图像在网页中位置的目的。通过表格布局的网页，网页中所有元素都放置在表格的单元格（<td>标记）中。

2.6.1　表格标记（<table>）

网页中的表格由<table>标记定义，一个表格被分成许多行<tr>，每行又被分成许多个单元格<td>，因此<table>、<tr>、<td>是表格中三个最基本的标记，它们必须同时出现才有意义。表格中的单元格能容纳网页中的任何元素，如图像、文本、列表、表单、表格等。

1. <table>标记

下面是一个最简单的表格代码，它的显示效果如图 2-21 所示。

```
<table border="1">
    <tr><td>CELL 1</td><td>CELL 2</td>
    </tr>
    <tr><td>CELL 3</td><td>CELL 4</td>
```

```
    </tr>
</table>
```

从图 2-21 可知，一个<tr>标记表示一行，<tr>标记中有两个<td>标记，表示一行中有两个单元格，因此显示为两行两列的表格。要注意在表格中行比列大，总是一行<tr>中包含若干个单元格<td>。

在这个表格<table>标记中还设置了边框宽度(border="1")，它表示表格的边框宽度是 1 像素。下面将边框宽度调整为 10 像素，即<table border="10">，这时显示效果如图 2-22 所示。

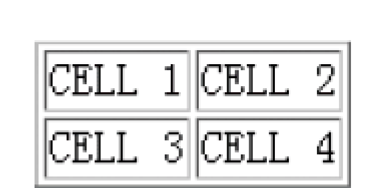


图 2-21　最简单的表格

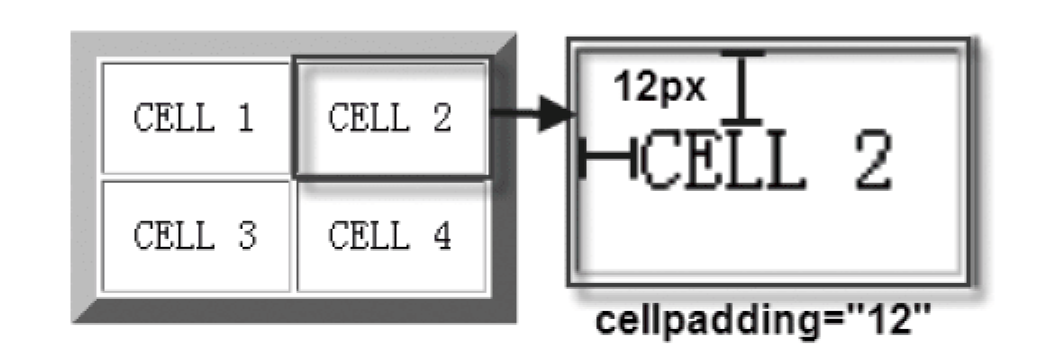


图 2-22　cellpadding 属性

此时虽然表格的边框宽度变成了 10 像素，但表格中每个单元格的边框宽度仍然是 1 像素，从这里可看出设置表格边框宽度不会影响单元格的边框宽度。

但有一个例外，如果将表格的边框宽度设置为 0，即<table border="0">(由于 border 属性的默认值就是 0，因此也可将 border 属性删除)，则单元格的边框宽度也跟着变为了 0。此时表格边框和单元格边框都消失，在用表格进行网页布局时通常需要这样设置。

由此可得出结论：设置表格边框为 0 时，会使单元格边框也变为 0；而设置表格边框为其他数值时，单元格边框宽度保持不变，始终为 1。

2. 填充(cellpadding)和间距(cellspacing)属性

填充(cellpadding)和间距(cellspacing)是<table>标记两个很重要的属性：cellpadding 表示单元格中的内容到单元格边框之间的距离，默认值为 0；而 cellspacing 表示相邻单元格之间的距离，默认值为 1。

合理设置填充和间距属性将美化表格。例如将表格填充设置为 12，即<table border="10" cellpadding="12">，则显示效果如图 2-22 所示。

把表格填充设置为 12，间距设置为 15，即<table border="10" cellpadding="12" cellspacing="15">，则显示效果如图 2-23 所示。

此外，表格<table>标记还具有宽(width)和高(height)、水平对齐(align)、背景颜色(bgcolor)等属性。表 2-7 列出了<table>标记的常见属性。

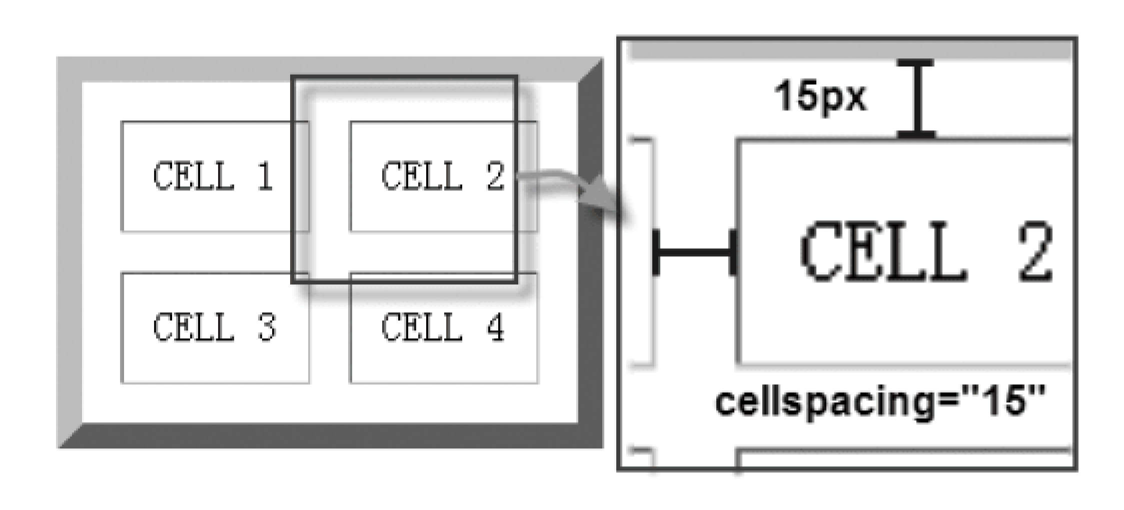


图 2-23　cellspacing 属性

表 2-7　<table>标记的属性

<table>标记的属性	含　义
border	表格边框的宽度，默认值为 0
bordercolor	表格边框的颜色，若不设置，将显示立体边框效果
bgcolor	表格的背景色
background	表格的背景图像
cellspacing	表格的间距，默认值为 1
cellpadding	表格的填充，默认值为 0
width，height	表格的宽和高，可以使用像素或百分比做单位
align	表格的水平对齐属性，默认为 left
rules	只显示表格的行边框或列边框

2.6.2　行<tr>和单元格<td>、<th>标记

<tr>表示表格中的一行，该标记有一些属性，如 align：统一设置该行中所有单元格的水平对齐方式，valign：统一设置该行中所有单元格的垂直对齐方式，bgcolor：设置该行的背景颜色。

表头标记<th>相当于一个特殊的单元格<td>标记，只不过<th>中的字符会以粗体居中方式显示。可以将表格第一行（第一个<tr>）中的<td>换成<th>，表示表格的表头。

对于单元格标记<td>、<th>来说，它们具有一些共同的属性，包括 width、height、align、valign、nowrap（不换行）、bordercolor、bgcolor 和 background。这些属性对于行标记<tr>来说，大部分都具有，只是没有 width 和 background 属性。

1. 单元格标记的对齐属性

单元格<td>或<th>标记具有 align 和 valign 属性，其含义如下：

（1）align——单元格中内容的水平对齐属性，取值有 left（默认值）、center、right。

（2）valign——单元格中内容的垂直对齐属性，取值有 middle（默认值）、top、bottom。

即单元格中的内容默认是水平左对齐，垂直居中对齐的。由于默认情况下单元格是

以能容纳内容的最小宽度和高度定义大小的，所以必须设置单元格的宽和高使其在大于最小宽高值时才能看到对齐的效果。例如，下面的代码的显示效果如图 2-24 所示。

```
<table width="256" border="4" cellpadding="2">
  <tr valign="bottom" height="58">
    <td width="82">底端对齐</td>
    <td width="96" valign="top">顶端对齐</td>
  </tr>
  <tr align="center" height="54">
    <td valign="top">水平居中顶端</td>
    <td>水平居中</td>
  </tr>
</table>
```



图 2-24　align 属性和 valign 属性

2．bgcolor 属性

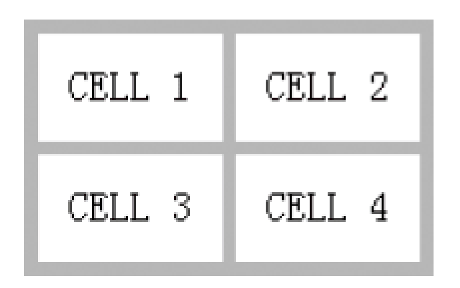
bgcolor 属性是<table>、<tr>、<td>都具有的属性，用来对表格或单元格设置背景色。在实际应用中，常将所有单元格的背景色设置为一种颜色，将表格的背景色设置为另一种颜色。此时如果间距(cellspacing)不为 0，则表格的背景色会环绕单元格，使间距看起来像边框一样。例如，下面的代码显示效果如图 2-25 所示。

```
<table border="1" cellpadding="12" cellspacing="5" bordercolor="#333333" bgcolor="#cccccc">
<tr bgcolor="#ffffff"><td>CELL 1</td><td>CELL 2</td></tr>
<tr bgcolor="#ffffff"><td>CELL 3</td><td>CELL 4</td></tr>
</table>
```

如果在此基础上将表格的 border 属性设置为 0，则显示效果如图 2-26 所示，可看出此时间距像边框一样了，而这个由间距形成的“边框”实际上是表格的背景色。

CELL 1	CELL 2
CELL 3	CELL 4

图 2-25　设置表格背景色为灰色、单元格背景色为白色的效果



CELL 1	CELL 2
CELL 3	CELL 4

图 2-26　在图 2-25 的基础上将表格边框宽度设置为 0

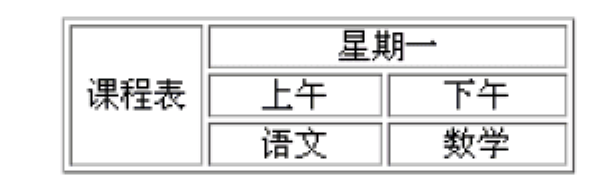
如果要减少 bgcolor 属性的书写次数，可以使用<tbody>标记，在所有<tr>标记的外面嵌套一个<tbody>标记，再设置<tbody>的背景色为白色即可，例如：

```
<table cellpadding="12" cellspacing="5" bordercolor="#333333" bgcolor=
"#cccccc">
    <tbody bgcolor="#ffffff">
      <tr><td>CELL 1</td><td>CELL 2</td></tr>
      <tr><td>CELL 3</td><td>CELL 4</td></tr>
    </tbody>
</table>
```

提示：<tbody>是表格体标记，它包含表格中所有的行或单元格。因此，如果所有单元格或行的某个属性都相同，可以将该属性写在<tbody>标记中，例如上述代码中的(bgcolor="#ffffff")，这样可减少代码冗余。

3. 单元格的合并属性

如果要合并某些单元格制作出如图 2-27 所示的表格，则必须使用单元格的合并属性，单元格<td>标记的合并属性有 colspan(跨多列属性)和 rowspan(跨多行属性)，是<td>标记特有的属性，分别用于合并列或合并行。例如：



课程表	星期一	
	上午	下午
	语文	数学

图 2-27　单元格合并后的效果

```
<td colspan="2">星期一</td>
```

表示该单元格由 2 列(2 个并排的单元格)合并而成，它将使该行<tr>标记中减少一个<td>标记。又如：

```
<td rowspan="3">课程表</td>
```

表示该单元格由 3 行(3 个上下排列的单元格)合并而成，它将使该行下的两行，两个<tr>标记中分别减少一个<td>标记。

如果一个单元格由 3 行 3 列 9 个单元格合并而成，则需要同时使用 colspan 和 rowspan 属性。

提示：设置了单元格合并属性后，再对单元格的宽或高进行精确设置会发现不容易了，因此在用表格布局时不推荐使用单元格合并属性，使用表格嵌套更合适些。

4. <caption>标记及其属性

<caption>标记用来为表格添加标题，这个标题固然可以用普通的文本实现，但是使用<caption>标记可以更好地描述这个表格的含义。

```
<table cellpadding="12" cellspacing="5" bgcolor="#cccccc">
    <caption>产品目录表</caption>      <!--<caption>必须位于<table>标记内-->
    <tr><td>……</td></tr>
</table>
```

2.6.3　在 Dreamweaver 中操作表格的方法

1. 在 Dreamweaver 中选中表格的方法

对表格进行操作之前必须先选中表格，有时几层表格嵌套在一起，使用以下方法仍然可以方便地选中表格或单元格。

(1) 选择整个表格：将鼠标指针移到的表格左上角或右下角时，光标右下角会出现表格形状，此时单击就可以选中整个表格，或者在表格区域内单击，再选择状态栏中的<table>标签按钮。

(2) 选择一行或一列单元格：将鼠标指针置于一行的左边框上，或置于一列的顶端

边框上，当选定箭头(⬇)出现时单击，选择一行也可单击状态栏中的<tr>标签按钮。

(3) 选择连续的几个单元格：在一个单元格中单击并拖动鼠标横向或纵向移至另一单元格。

(4) 选择不连续的几个单元格：按住 Ctrl 键，单击欲选定的单元格、行或列。

(5) 选择单元格中的网页元素：直接单击单元格中的网页元素。

提示：*按住 Ctrl 键，鼠标在表格上滑动，Dreamweaver 会高亮显示表格结构。*

2. 向表格中插入行或列的方法

当光标位于表格内时，在右键快捷菜单中选择"表格"→"插入行(或插入列)"命令，可在表格的当前行的上方插入一行，或当前行的左边插入一列；若要在表格的最右边插入一列或最下方插入一行，可选择"表格"→"插入行或列…"命令，即在所选列之后或所选行之下插入列或行。插入行也可以在代码视图中复制一行的代码"<tr>…</tr>"再粘贴几次就插入了几行，而插入列在代码视图中不方便进行。

3. 设置单元格中内容居中对齐的方法

在默认情况下，表格会单独占据网页中的一行，左对齐排列。表格具有水平对齐属性 align，可以设置 align="center"让表格水平居中对齐，位于一行的中央。而单元格<td>则具有水平对齐 align 和垂直对齐 valign 属性，它们的作用是使单元格中的内容相对于单元格水平居中或垂直居中，在默认情况下，单元格中的内容是垂直居中，但水平左对齐的。

如果在单元格中有一段无格式的文字，例如：

```
<td>版权所有 &copy;数学系</td>
```

(1) 要使这段文字在单元格中居中对齐，那么有两种方法可以做到。一是在设计视图中选中这些文字，然后使用文本自身的对齐属性来居中对齐。即单击图 2-28 中①处的按钮。



图 2-28　单元格中文本对齐的两种方法

此时，可发现文本已经居中，切换到代码视图，代码已修改为：

```
<td><div align="center">版权所有 &copy;数学系</div></td>
```

可看到使用这种方法对齐 Dreamweaver 会自动为文本添加一个 div 标记，再使用 div 标记的 align 属性使文本对齐，这是因为这段文本没有格式标记环绕，要使它们居中只能添加一个标记，如果这段文本被格式标记环绕，例如 p 标记，那么就会直接在 p 标记中添加 align="center"属性了。

(2) 由于文本位于单元格中，第二种使文本居中的方法是利用单元格的居中对齐属性，即单击图 2-28 中②处的按钮，可发现文本也能居中对齐，在代码视图查看代码如下：

```
<td align="center">版权所有 &copy;数学系</td>
```

可看到第二种方法不会增加一个标记，所以推荐使用这种方法对齐单元格中的文本。

2.6.4 制作固定宽度的表格

如果不定义表格中每个单元格的宽度，当向单元格中插入网页元素时，表格往往会变形。这样无法利用表格精确定位网页中的元素，网页中会有很多不必要的空隙，使网页显得不紧凑也不美观，因此要利用固定宽度的表格和单元格精确地包围住其中的内容。制作固定宽度的表格通常有以下两种方法：

(1) 定义所有列的宽度，但不定义整个表格的宽度。例如：

```
<table border="0" cellspacing="0" cellpadding="0">
  <tr>
    <td width="200"> </td>
    <td width="360"> </td>
    <td width="200"> </td>
</tr></table>
```

整个表格的实际宽度为：所有列的宽度和＋边框宽度和＋间距和＋填充和。这时候，只要单元格内的内容不超过的单元格的宽度时，表格不会变形。

(2) 定义整个表格的宽，如 500 像素、98％等，再留一列的宽度不定义，未定义的这一列的宽度为整个表格的宽度－已定义列的宽度和－边框宽度和－间距和－填充和，同样在插入内容时也不会变形。

由于网页的总宽度、每列的宽度都要固定，所以制作固定宽度的表格是用表格进行网页布局的基础。而网页布局时一般是不需要指定布局表格高度的，因为随着单元格中内容的增加，布局表格的高度也会自适应的增加。

因此制作固定高度的表格相对来说用得较少，只有在单元格中插入图像时，为了保证单元格和图像之间没有间隙，需要把单元格的宽和高设置为图像的宽和高，填充、间距和边框值都设为 0，并保证单元格标记内除开图像元素，没有其他空格或换行符。

提示：在用表格布局时不推荐使用鼠标拖动表格边框的方式来调整其大小，这样会在表格标记内自动插入 width 和 height 属性。如果所有单元格的宽已固定，又定义了表格的宽度，那么所有单元格的宽度都会按比例发生改变，导致用表格布局的网页里的内容排列混乱。

2.6.5 特殊效果表格的制作

1. 制作 1 像素(细线)边框的表格

一般来说，1 像素边框的表格在网页中显得更美观。特别是用表格做栏目框时，1 像素边框的栏目框是大部分网站的选择，因此，制作 1 像素边框的表格已成为网页设计的一

项基本要求。

但是把表格的边框(border)定义为 1 像素时(border="1"),其实际宽度是 2 像素。这样的表格边框显得很粗而不美观。要制作 1 像素的细线边框可用如下任意一种方法实现。

(1) 用间距做边框。原理是通过把表格的背景色和单元格的背景色调整成不同的颜色,使间距看起来像一个边框一样,再将表格的边框设为 0,间距设为 1,即实现 1 像素"边框"表格。代码如下:

```
<table border="0" cellspacing="1" bgcolor="#000000">
    <tr><td bgcolor="#ffffff">1像素边框表格</td></tr>
</table>
```

(2) 用 CSS 属性 border-collapse 做 1 像素边框的表格。先把表格的边框(border)设为 1,间距(cellspacing)设为 0,此时表格的边框和单元格的边框紧挨在一起,所以边框的宽度为 1+1=2 像素。这是因为表格的 CSS 属性 border-collapse 的默认值是 separate,即表格边框和单元格边框不重叠。当我们把 border-collapse 属性值设为 collapse(重叠)时,表格边框和单元格边框将发生重叠,因此边框的宽度为 1 像素。代码如下:

```
<table border="1" cellspacing="0" bordercolor="#ff0000" style="border-collapse: collapse">…</
table>
```

2. 制作双线边框表格

将表格的边框颜色(bordercolor)属性设置为某种颜色后,表格的暗边框和亮边框会变为同一种颜色(在 IE 中单元格边框的颜色也会跟着改变),边框的立体感消失。此时只要间距(cellspacing)不设为 0,表格的边框和单元格的边框就不会重合,如果设置表格的边框宽度为 1 像素,则显示为双细线边框表格。下面是用双细线边框表格制作的栏目框,效果如图 2-29 所示。

```
<table width="180" border="1" cellpadding="6" cellspacing="3" bordercolor=
"#000000" bgcolor="#ffffff">
    <tr><td bgcolor="#cccccc">标题</td></tr>
    <tr><td height="128" valign="top" bordercolor="#ffffff">内容</td>
    </tr>
</table>
```

由于非 IE 浏览器无法改变单元格边框的颜色,因此这种双线边框栏目框只能在 IE 8 以下的版本中看到效果。

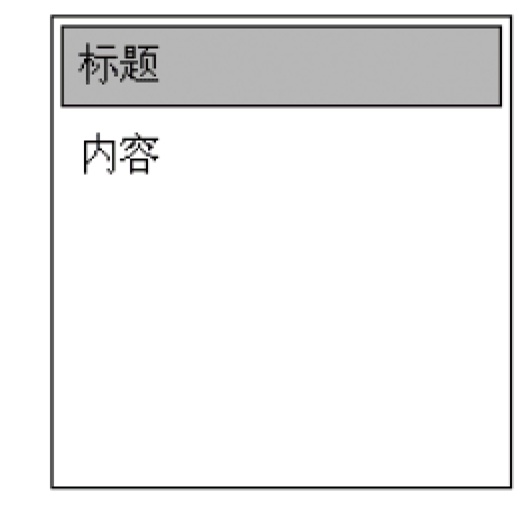


图 2-29 IE 中双线边框栏目框

3. 用单元格制作水平线或占位表格

如果需要水平或竖直的线段,可以使用表格的行或列来制作,例如在表格中需要一条黑色的水平线段,则可以这样制作:先把某一行的行高设为 1;再把该行的背景色

设为黑色；最后在代码视图中去掉此行单元格中的" "占位符空格。因为" "是 Dreamweaver 在插入表格时自动往每个单元格中添加的一个字符；如果不去掉，IE 默认一个字符占据 12 像素的高度。这样就制作了一条 1 像素粗的水平黑线。代码如下：

```
<table width="200" border="0" cellpadding="0" cellspacing="0">
    <tr><td height="1" bgcolor="#000000"></td>  <!--单元格中的" "已去掉-->
    </tr>
</table>
```

如果要制作 1 像素粗的竖直黑线，可在上述代码中将表格的宽修改为 1 像素，单元格的高修改为竖直黑线的长度即可。

在默认情况下，网页中两个相邻的表格上下会紧挨在一起，这时可以在这两个表格中插入一个占位表格使它们之间有一些间隙，例如把占位表格的高度设置为 7 像素，边框、填充、间距设为 0，并去掉单元格中的" "，则在两个表格间插入了一个 7 像素高的占位表格，这样就避免了表格紧挨的情况出现，因为我们通常都不希望两个栏目框上下紧挨在一起。当然，通过对表格设置 CSS 属性 margin 能更容易地实现留空隙。

4. 用表格制作圆角栏目框

上网时经常可以看到漂亮的圆角栏目框，下面就来制作一个固定宽度的圆角栏目框，如图 2-30 所示。由于表格只能是一个矩形，所以制作圆角的原理是在圆角部分插入圆角图片。制作步骤如下：

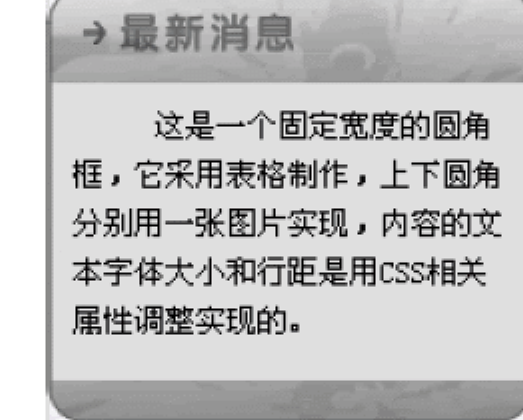


图 2-30 用表格制作的圆角栏目框

(1) 准备两张圆角图片，分别是上圆角和下圆角的图像。

(2) 插入一个三行一列的表格，把表格的填充、间距和边框设为 0，宽设置为 190 像素(圆角图片的宽)，高不设置。

(3) 分别设置表格内三个单元格的高。第一个单元格高设置为 38 像素(上圆角图片的高)；第二个单元格高为 100 像素；第三个单元格高为 17 像素(下圆角图片的高)。在第 1 和第 3 个单元格内分别插入上圆角和下圆角的图片。

(4) 把第二个单元格内容的水平对齐方式设置为居中(align="center")，单元格的背景颜色设置为圆角图片边框的颜色(bgcolor="#E78BB2")。

(5) 这时在第二个单元格内再插入一个一行一列的表格，把该表格的间距和边框设为 0，填充设为 8 像素(让栏目框中的内容和边框之间有一些间隔)，宽设为 186 像素，高为 100 像素。背景颜色设置为比边框浅的颜色(bgcolor="#FAE4E6")。

提示：第(5)步也可以不插入表格，而是把第二个单元格拆分成 3 列，把 3 列对应的 3 个单元格的宽分别设置为 2 像素、186 像素和 2 像素，并在代码视图把这 3 个单元格中的" "去掉，然后把第 1 和第 3 列的背景色设置为圆角边框的颜色，第 2 列的背景色设为圆角背景的颜色，并用 CSS 属性设置它的填充为 8 像素(style="padding:8px")。

2.6.6 用表格进行网页布局

1-3-1 版式布局是一种最常用的网页版面布局方式，也是学习其他复杂版面布局的基础，它可以通过画 4 个表格实现，如图 2-31 所示。下面分别用普通表格和布局模式下的表格来实现如图 2-31 所示的布局。

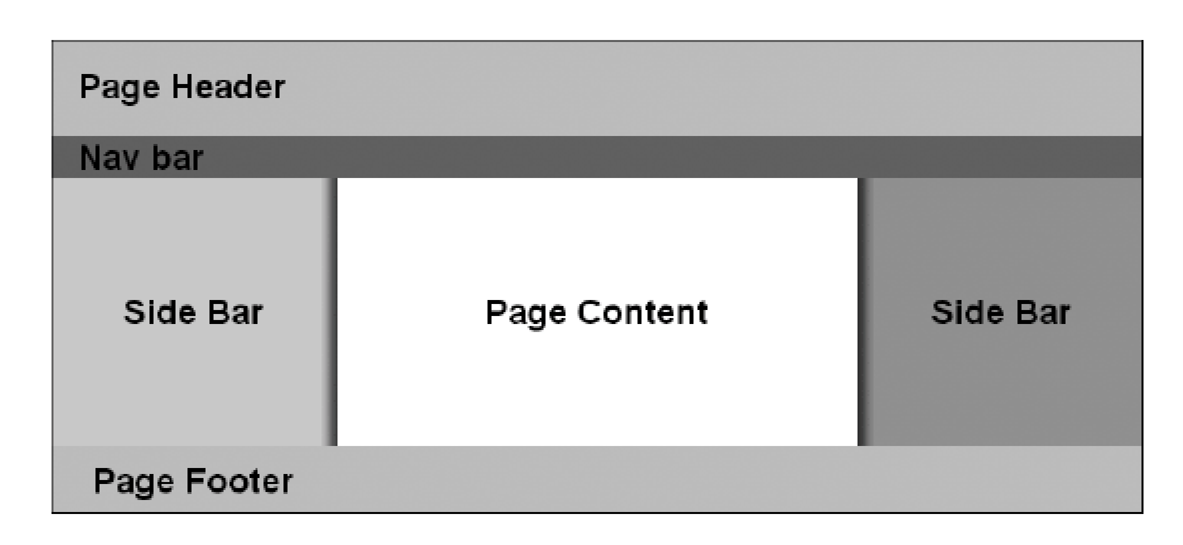


图 2-31 1-3-1 版面布局

用表格进行网页布局制作如图 2-31 所示网页的步骤如下：

(1) 单击“常用”工具栏中的“表格”按钮，插入一个一行一列的表格，该表格用于放置 Page Header，将表格宽度设置为 768 像素，边框、单元格边距和单元格间距都设置为 0，其他地方保持默认，如图 2-32 所示。



图 2-32 插入表格对话框

实际上也可以不在这里设置表格参数，等插入表格之后，再选中该表格在它的属性面板里进行设置。但该表格对话框具有记忆功能，以后每次插入表格时都会默认显示前一次设置的值。因为后面还要插入几个宽度相同，边框、边距、间距都为 0 的表格，所以还是在这里设置简便些。

(2) 以同样的方式再插入一个一行一列的表格，该表格用于放导航条(Nav bar)。

(3) 接着再插入一个一行三列的表格，该表格用于放网页内容的主体，将左边单元格和右边单元格的宽度均设置为 200 像素，中间一列不设置宽度。然后在属性面板中将三个单元格的“垂直”对齐方式均改为顶端对齐，切换到代码视图可看到三个单元格的<td>标记中均添加了一个属性 valign="top"。这样三列中的内容就会自动从上到下排列，接下来可以为左右两栏的单元格设置背景颜色。

(4) 最后再插入一个一行一列的表格，用于放置 Page Footer 部分。

2.6.7 表格布局实例——制作太阳能网站

本节介绍用表格布局的方法制作某太阳能公司网站，由于太阳能热水器是一种绿色

环保产品,因此该公司网页以绿色为主色调,采用深绿色和黄绿色搭配的同类色配色方案设计,整体效果如图 2-33 所示。网页的布局表格结构如图 2-34 所示。



图 2-33　太阳能公司网站首页整体效果

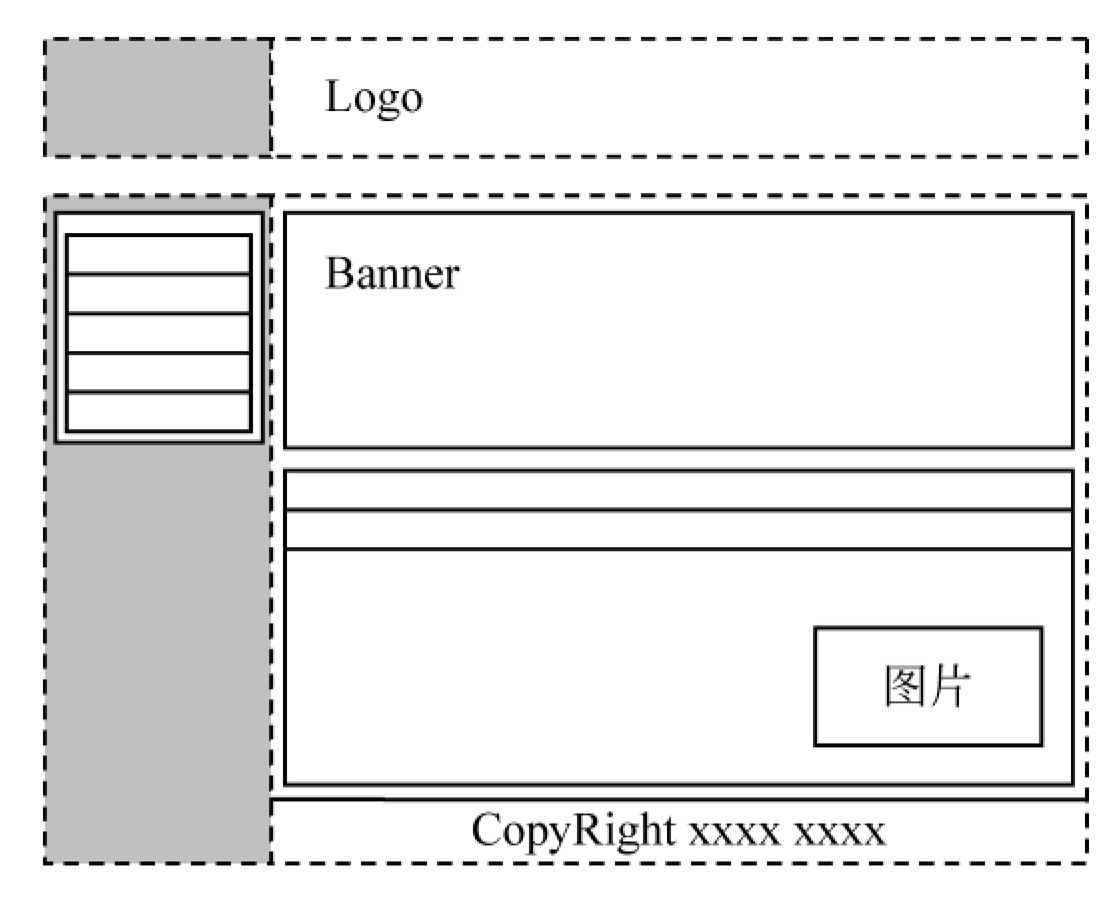


图 2-34　网页表格布局的结构图

可以看出,该网页头部采用一个一行两列的表格,主体部分也是一个一行两列表格,以便将它分成两栏,这样左侧栏和右侧栏中的内容不会相互产生影响。

1. 制作网页的头部

(1) 插入一个一行两列的表格，将宽设为852像素，高设为104像素，并将表格边框、单元格边距(填充)、单元格间距均设置为0。(说明，本网页中的所有表格都是用作布局表格，布局表格通常都要将表格边框、填充、间距设为0，以下的表格如无特殊说明也需这样设置)。

(2) 将左边单元格宽设为161像素，背景色设为#99cc00。

(3) 在右边单元格插入一张图片images/logo.jpg。

(4) 可看到图片位于右边单元格的最左边，为了使图片向右边移一些，在该单元格左边插入一个单元格，方法是在该单元格中右击，选择"表格"→"插入行或列"→"插入列"→"当前列之前"菜单命令，设置宽为64像素，此时图片向右移动了64个像素，可看到此单元格起到了一个占位的作用。

2. 制作网页的主体部分

(1) 在网页头部表格下插入一个一行两列的表格，将宽设为852像素，这样就将网页主体部分分为左右两栏。将两个单元格都设置为垂直顶端对齐(valign="top")(布局单元格通常都应有此设置，使得其中的内容能从顶端开始往下排列)。

(2) 制作左侧栏部分，将左侧单元格宽设为161像素，高设为617像素(网页制作完成后可将该高度属性去掉)，背景色设为#99cc00。在左侧栏中插入一个一行一列的表格，作为导航栏的背景，宽设置为100%，高设置为181像素，将其单元格的背景色设为深绿色(#00801b)。

(3) 在该表格中插入一个六行一列的表格用于放置导航按钮，将宽设为143像素(和导航图片等宽)，将表格设为中对齐。

(4) 在该表格的每个单元格中分别插入一个导航图片(dh1.jpg-dh6.jpg，本实例中所有图片均位于images文件夹下)，并分别将这些图片先链接到"#"(空链接)。

(5) 制作右侧栏中的Banner。在网页主体表格的右侧单元格中插入一个一行一列的表格，设置表格宽为688像素，高为181像素。将表格的背景图像设置为images/ba1.jpg(注意，此处一定要将图片作为表格的背景放入，而不能插入图片，否则无法在其上面再叠放Flash文件)。

(6) 制作"公司简介"栏目。首先插入一个三行一列的表格，将宽设置为90%，将表格设置为居中对齐。

(7) 设置第一个单元格高为41像素，设置背景图像为images/bj.jpg，背景图像默认会平铺满单元格，再在该单元格中插入一幅图像images/ggd.jpg，可看到图像和背景图像很好地融合在了一起，如图2-35所示。

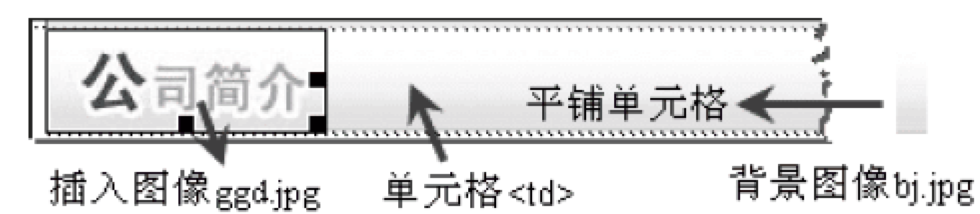


图2-35 在单元格中同时使用图像和背景图像

(8) 设置第二个单元格高为21像素，起占位的作用。

(9) 在第三个单元格中插入一段公司简介的文本。

(10) 在文本中间插入一幅客服的图像 images/in.jpg，为了使该图像能够被文字环绕，设置该图像的对齐方式为右对齐(align="right")。

3. 完善网页及插入 Flash

(1) 现在网页主体已基本呈现，下面进行一些微调。可看到此时公司简介栏目表格与 banner 表格紧挨在一起，不美观。我们在两者之间插入一个占位表格(方法是将光标移动到公司简介栏目表格的左边)，设置该表格宽为 100%，高为 16 像素。

(2) 制作网页底部版权部分。在公司简介表格下插入一条水平线，方法是将工具栏切换到 HTML，插入"水平线"，再在属性面板中设置其宽为 90%，高为 1 像素，居中对齐，无阴影。再切换到代码视图，对<hr/>标记设置属性(color="gray")以改变水平线的颜色。再在下面插入一段版权说明文本，设置该文本为居中对齐。

(3) 插入 Flash。在放置网页 banner 的单元格中插入一幅 Flash(images/ba.swf)，执行"插入"→"媒体"→Flash 菜单命令。在"属性"面板中设置 Flash 宽为 400 像素，高为 100 像素。再选中该 Flash，单击"参数"按钮，设置参数 wmode 值 transparent，使 Flash 能透明显示。

(4) 调整该 Flash 在 banner 上的位置，方法是选中放置 banner 图片的表格，设置该表格的填充值为 28 像素，这样 Flash 与单元格左边会有 28 像素的距离。

(5) 用 CSS 设置网页主体的右边框线(方法是在网页主体表格的右侧单元格<td>标记中加入属性 style=" border-right：#daeda3 1px solid")。

(6) 为了改变网页中所有字体的大小、颜色、行高，必须用 CSS 设置文本样式，在网页头部区域加入"<style>td{font-size：9pt;line-height：18pt；color：#333;}</style>"即可。

提示：如果要在当前表格上方插入一个表格，可将光标移动到该表格左侧再插入表格；如果要在当前表格下方插入一个表格，可将光标移动到该表格右侧再插入表格。

2.7 创建表单

表单是浏览器与服务器之间交互的重要手段，利用表单可以收集客户端提交的有关信息。例如如图 2-36 所示的是用户注册表单，用户单击"提交"按钮后表单中的信息就会发送到服务器。

表单由表单界面和服务器端程序(如 PHP)两部分构成。表单界面由 HTML 代码编写，服务器端程序用来收集用户通过表单提交的数据。本节只讨论表单界面的制作。在 HTML 代码中，可以用表单标记定义表单，并且指定接收表单数据的服务器端程序文件。

表单处理信息的过程为：当单击表单中的"提交"按钮时，在表单中填写的信息就会发送到服务器，然后由服务器端的有关应用程序进行处理，处理后或者将用户提交的信息存储在服务器端的数据库中，或者将有关的信息返回到客户端浏览器。

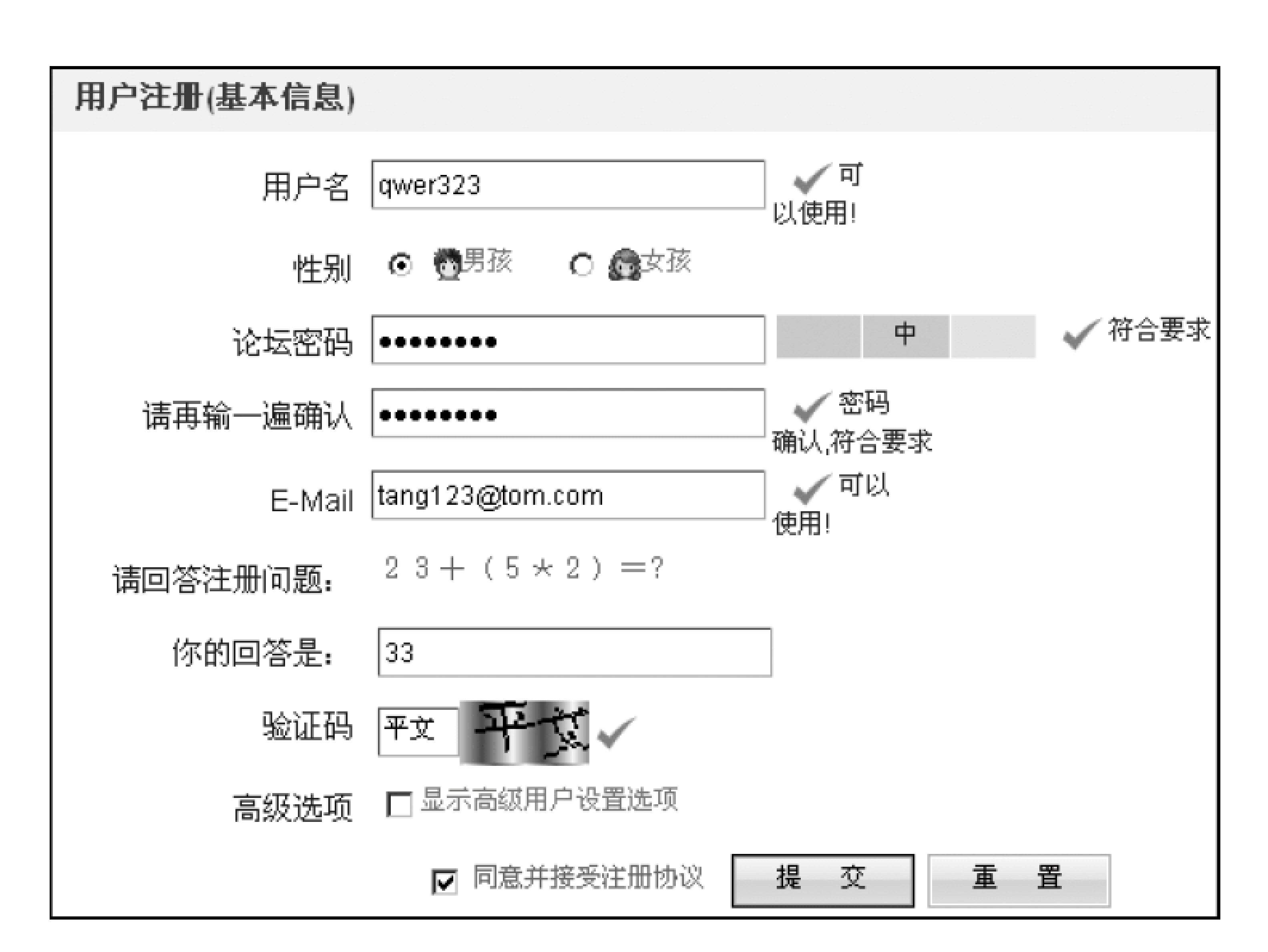


图 2-36 用户注册表单

2.7.1 ＜form＞标记及其属性

＜form＞标记用来创建一个表单，即定义表单的开始和结束位置，这一标记有几方面的作用。首先，限定表单的范围，一个表单中的所有表单域标记，都要写在＜form＞与＜/form＞之间，单击"提交"按钮时，提交的也是该表单范围内的内容。其次，携带表单的相关信息，例如处理表单的脚本程序的位置(action)、提交表单的方法(method)等，这些信息对于浏览者是不可见的，但对于处理表单却起着决定性的作用。

＜form＞标记中包含的表单域标记通常有＜input＞、＜select＞和＜textarea＞等。图 2-37 展示了 Dreamweaver 的表单工具栏中各种表单元素与标记的对应关系。

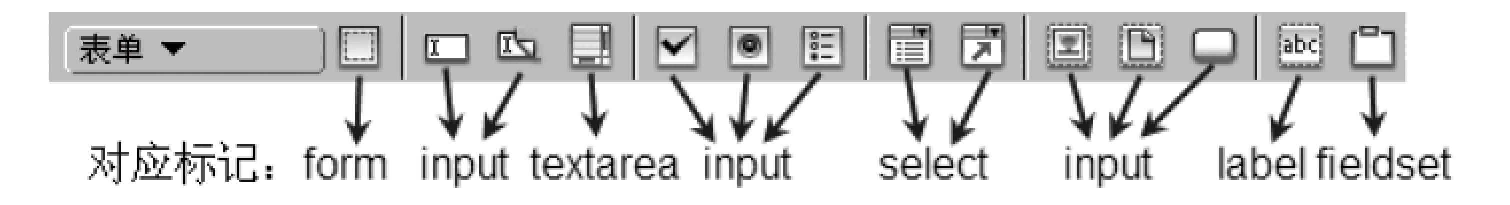


图 2-37 表单元素和表单标记的对应关系

在图 2-37 中单击"表单"按钮()后，就会在网页中插入一个表单＜form＞标记，此时会在"属性"面板中显示＜form＞标记的属性设置，如图 2-38 所示。

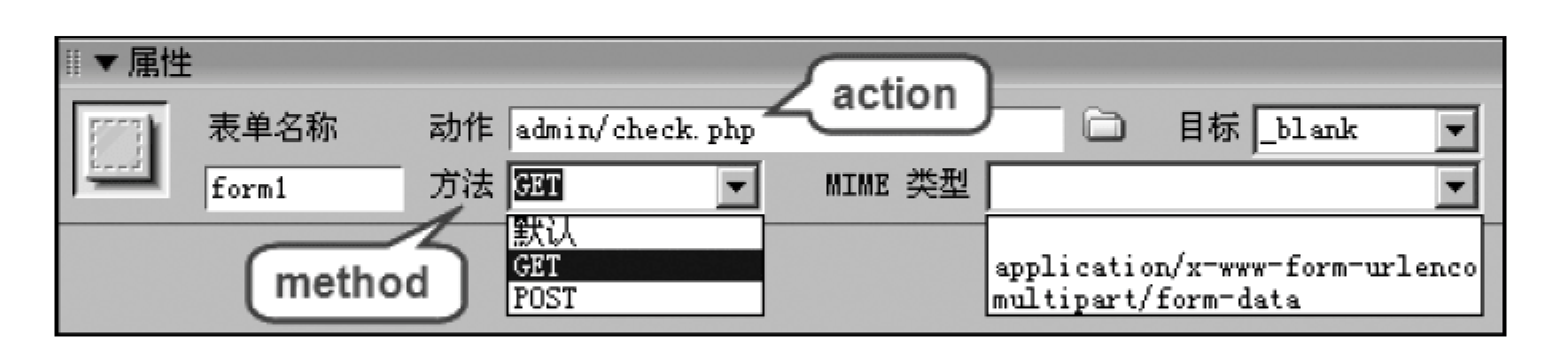


图 2-38 ＜form＞标记的"属性"面板

＜form＞标记具有的属性如下：

1. name 属性

在图 2-38 中,"表单名称"对应 name 属性,可设置一个唯一的名称以标识该表单,如<form name="form1">,该名称仅供 JavaScript 代码调用表单中的元素。

2. action 属性

"动作"对应表单的 action 属性。action 属性用来设置接收表单内容的程序文件的 URL。例如,<form action="admin/check.php">,表示当用户提交表单后,将转到 admin 目录下的 check.php 页面,并由 check.php 接收发送来的表单数据,该文件执行完毕后(通常是对表单数据进行处理),将返回执行结果(生成的静态页)给浏览器。

在"动作"文本框中可输入相对 URL 或绝对 URL。如果不设置 action 属性(即 action=""),表单中的数据将提交给表单自身所在的文件,这种情况常见于将表单代码和处理表单的程序写在同一个动态网页中的情况,否则将没有接收和处理表单内容的程序。

3. method 属性

"方法"对应<form>的 method 属性,定义浏览器将表单数据传递到服务器的方式。取值只能是 GET 或 POST(默认值是 GET)。例如:

```
<form method="post">
```

(1) 使用 GET 方式时,Web 浏览器将各表单字段名称及其值按照 URL 参数格式的形式,附在 action 属性指定的 URL 地址后一起发送给服务器。例如,一个使用 GET 方式的 form 表单提交时,在浏览器地址栏中生成的 URL 具有类似下面的形式:

```
http://ec.hynu.cn/admin/check.php?name=alice&password=123
```

GET 方式生成的 URL 格式为:每个表单域元素名称与取值之间用等号"="分隔,形成一个参数;各个参数之间用"&"分隔;而 action 属性所指定的 URL 与参数之间用问号"?"分隔。

(2) 使用 POST 方式时,浏览器将把各表单域元素名称及其值作为 HTTP 消息的实体内容发送给 Web 服务器,而不是作为 URL 参数传递。因此,使用 POST 方式传送的数据不会显示在地址栏中。

提示:不要使用 GET 方式发送大数据量的表单(例如,表单中有文件上传域时)。因为 URL 长度最多只能有 8192 个字符,如果发送的数据量太大,数据将被截断,从而导致发送的数据不完整。另外,在发送机密信息时(如用户名和口令、信用卡号等),不要使用 GET 方式;否则,浏览者输入的口令将作为 URL 显示在地址栏上,而且还将保存在浏览器的历史记录文件和服务器的日志文件中。因此,GET 方式不适合于发送有机密性要求的数据和发送大数据量数据的场合。

4. enctype 属性

“MIME 类型”对应<form>的 enctype 属性，用来指定表单数据在发送到服务器之前应该如何编码。默认值为“application/x-www-form-urlencode”，表示表单中的数据被编码成“名＝值”对的形式，因此在一般情况下无须设置该属性。但如果表单中含有文件上传域，则需设置该属性为“multipart/form-data”，并设置提交方式为 POST。

5. target 属性

“目标”对应<form>的“target”属性，它指当提交表单时，action 属性所指定的网页以何种方式打开（在新窗口还是原窗口），取值有 4 种，其含义和<a>标记的 target 属性相同（见表 2-6）。

2.7.2 <input/>标记

<input/>标记是用来收集用户输入信息的标记，它是一个单标记，<input/>至少应具有两个属性：一是 type 属性，用来决定这个<input>标记的含义，type 属性共有 10 种取值，各种取值的含义如表 2-8 所示；二是 name 属性，用来定义该表单域标记的名称，如果没有该属性，虽然不会影响表单的界面，但服务器将无法获取该表单域提交的数据。

表 2-8 <input>标记的 type 属性取值含义

type 属性值	含　义	type 属性值	含　义
text	文本框	submit	提交按钮
password	密码框	reset	重置按钮
radio	单选框	button	普通按钮
checkbox	复选框	image	图像按钮
file	文件域	hidden	隐藏域

1. 单行文本框

当<input/>的 type 属性为 text 时，即：<input type="text" …/>，将在表单中创建一个单行文本框，如图 2-39 所示。文本框用来收集用户输入的少量文本信息。例如：

```
姓名：<input type="text" name="user" size="20" />
```

表示该单行文本框的宽度为 20 个字符，名称属性为 user。

如果用户在该文本框中输入了内容（假设输入的是 Tom），那么提交表单时，提交给服务器的数据就是“user＝Tom”，即表单提交的数据总是“name＝value”对的形式。由于 name 属性值为 user，而文本框的 value 属性值为文本框中的内容，因此有以上结果。如果用户没有在该文本框中输入内容，那么提交表单时，提交给服务器的数据就是“user＝”。

在初次打开网页时文本框一般是空的。如果要使文本框显示初始值，可设置其 value

属性，value 属性的值将作为文本框的初始值显示。如果希望单击文本框时清空文本框中的值，可对 onfocus 事件编写 JavaScript 代码，因为单击文本框时会触发文本框的 onfocus 事件。示例代码如下，效果如图 2-39 所示。文本框和密码框的常用属性如表 2-9 所示。

```
查询<input type="text" name="seach" value="请输入关键字" onfocus="this.value=
''" />
```

查询 [请输入关键字]　　查询 [　　]

图 2-39　设置了 value 属性值的文本框在网页载入时（左）和单击后（右）

表 2-9　文本框和密码框的常用属性

属性名	功　能	示　例
value	设置文本框中的初始内容，如不设置，则文本框显示的初始内容为空。用户输入的内容将会作为 value 属性的最终值	value="请在此输入"
size	指定文本框的宽度，以字符个数为度量单位	size="16"
maxlength	设置用户能够输入的最多字符个数	maxlength="11"
readonly	文本框为只读，用户不能改变文本框中的值，但用户仍能选中或复制其文本，其内容也会发送给服务器	readonly="readonly"
disabled	禁用文本框，文本框将不能获得焦点，提交表单时，也不会将文本框的名称和值发送给服务器	disabled="disabled"

提示：readonly 可防止用户对值进行修改，直到满足某些条件为止（比如选中了一个复选框），此时需要使用 JavaScript 清除 readonly 属性。disabled 可应用于所有表单元素。

2. 密码框

当<input/>的 type 属性为 password 时，表示该<input/>是一个密码框。密码框和文本框基本相同，只是用户输入的字符会以圆点显示，以防被旁人看到。但表单发送数据时仍然会把用户输入的真实字符作为其 value 值以不加密的形式发送给服务器。示例代码如下，显示效果如图 2-40 所示。

密码：[●●●●●●]

图 2-40　密码框

```
密码：<input type="password" name="pw" size="15" />
```

3. 单选按钮

<input type="radio" …/>用于在表单上添加一个单选按钮，但单选按钮需要成组使用才有意义。只要将多个单选按钮的 name 属性值设置为相同，它们就形成一组单选按钮。浏览器只允许一组单选按钮中的一个被选中。当用户提交表单时，在一个单选按钮组中，只有被选中的那个单选按钮的名称和值（即 name/value 对）才会被发送给服务器。

因此同组的每个单选按钮的 value 属性值必须各不相同，以实现选中不同的单选项，

就能发送同一 name 不同 value 值的功能。下面是一组单选按钮的代码，效果如图 2-41 所示。

```
性别：男<input type="radio" name="sex" value="1" checked="checked" />
    女<input type="radio" name="sex" value="2" />
```

其中，checked 属性设定初始时单选按钮哪项处于选定状态，不设定表示都不选中。

4. 复选框

<input type="checkbox" />用于在表单上添加一个复选框。复选框可以让用户选择一项或多项内容，复选框的一个常见属性是 checked，该属性用来设置复选框初始状态时是否被选中。复选框的 value 属性只有在复选框被选中时才有效。如果表单提交时，某个复选框是未被选中的，那么复选框的 name 和 value 属性值都不会传递给服务器，就像没有这个复选框一样。只有某个复选框被选中，它的名称(name 属性值)和值(value 属性值)才会传递给服务器。下面的代码是一个复选框的例子，显示效果如图 2-42 所示。

```
爱好：<input name="fav1" type="checkbox" value="1" />跳舞
    <input name="fav2" type="checkbox" value="2" />散步
    <input name="fav3" type="checkbox" value="3" />唱歌
```

性别： 男 ⦿ 女 ○

图 2-41 单选按钮

爱好： ☐ 跳舞 ☐ 散步 ☐ 唱歌

图 2-42 复选框

提示：从以上示例可看出，选择类表单标记(单选框、复选框或下拉列表框等)和输入类表单标记(文本域、密码域、多行文本域等)的重要区别是：选择类标记必须事先设定每个元素的 value 属性值，而输入类标记的 value 属性值一般由用户输入，可以不设定。

5. 文件上传域

<input type="file" …/>是表单的文件上传域，用于浏览器通过表单向服务器上传文件。使用<input type="file" />元素，浏览器会自动生成一个文本框和一个“浏览…”按钮，供用户选择上传到服务器的文件，示例代码如下，效果如图 2-43 所示。

浏览...

图 2-43 文件上传域

```
<input type="file" name="upfile" />
```

用户可以使用“浏览…”按钮打开一个文件对话框，选择要上传的文件，也可以在文本框中直接输入本地的文件路径名。

注意：如果<form>标记中含有文件上传域，则<form>标记的 enctype 属性必须设置为“multipart/form-data”，并且 method 属性必须是 post。

6. 隐藏域

<input type="hidden" …/>是表单的隐藏域，隐藏域不会显示在网页中，但是当提

交表单时，浏览器会将这个隐藏域元素的 name/value 属性值对发送给服务器。因此隐藏域必须具有 name 属性和 value 属性，否则毫无作用。例如：

```
<input type="hidden" name="user" value="Alice" />
```

隐藏域是网页之间传递信息的一种方法。例如，假设网站的用户注册过程由两个步骤完成，每个步骤对应一个网页文件。用户在第一步的表单中输入了用户名，接着进入第二步的网页中，在这个网页中填写爱好和特长等信息。在第二个网页提交时，要将第一个网页中收集到的用户名也传送给服务器，这就需要在第二个网页的表单中加入一个隐藏域，让它的 value 值等于接收到的用户名。

2.7.3 <select>和<option>标记

<select>标记表示下拉框或列表框，是一个标记的含义由其 size 属性决定的元素。如果该标记没有设置 size 属性，那么就表示是下拉列表框。如果设置了 size 属性，则变成了列表框，列表的行数由 size 属性值决定。如果还设置了 multiple 属性，则表示列表框允许多选。下拉列表框中的每一项由<option>标记定义，还可使用<optgroup>标记添加一个不可选中的选项，用于给选项进行分组。例如，下面代码的显示效果如图 2-44 所示。

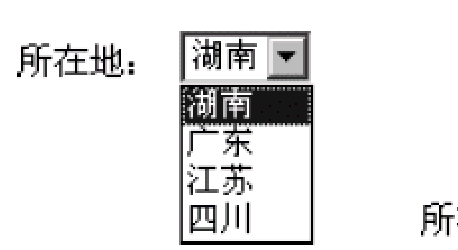


图 2-44 下拉列表框(左)和列表框(右)

```
所在地：<select name="addr">   <!--添加属性 size="5"则为图 2-44 右边的列表框-->
        <option value="1">湖南</option>
        <option value="2">广东</option>
        <option value="3">江苏</option>
        <option value="4">四川</option></select>
```

提交表单时，select 标记的 name 值将与选中项的 value 值一起作为 name/value 信息对传送给服务器。如果<option>标记没有设置 value 属性，那么提交表单时，将把选中项中的文本(例如“湖南”)作为 name/value 信息对的 value 部分发送给服务器。

2.7.4 多行文本域标记<textarea>

<textarea>是多行文本域标记，用于让浏览者输入多行文本，如发表评论或留言等。<textarea>是一个双标记，它没有 value 属性，而是将标记中的内容显示在多行文本框中，提交表单时也是将多行文本框中的内容作为 value 值提交。例如：

```
<textarea name="comments" cols="40" rows="4" wrap="virtual">表示是一个有 4行,每行可容纳 40个
字符,换行方式为虚拟换行的多行文本域。</textarea>
```

<textarea>的属性有：

(1) cols——用来设置文本域的宽度，单位是字符。

(2) rows——用来设置文本域的高度(行数)。

(3) wrap——设置多行文本的换行方式，默认值为 virtual，其取值有三种，含义如下：

① 关(off)——不让文本换行。当用户输入的内容超过文本区域的右边界时，文本将

向左侧滚动，不会换行。用户必须按 Enter 键才能将插入点移动到文本区域的下一行。

② 虚拟(virtual)——表示在文本区域中设置自动换行。当用户输入的内容超过文本区域的右边界时，文本换行到下一行。当提交数据进行处理时，换行符并不会添加到数据中。

③ 实体(physical)——文本在文本域中也会自动换行，但是当提交数据进行处理时，会把这些自动换行符转换为
标记添加到数据中。

2.7.5 表单数据的传递过程

1. 表单向服务器提交的信息内容

当单击表单的“提交”按钮后，表单将向服务器发送表单中填写的信息，发送形式是各个表单元素的“name=value & name=value & name=value…”。下面以图 2-45 中的表单为例来分析表单向服务器提交的内容(输入的密码是 123)，该表单的代码如下：

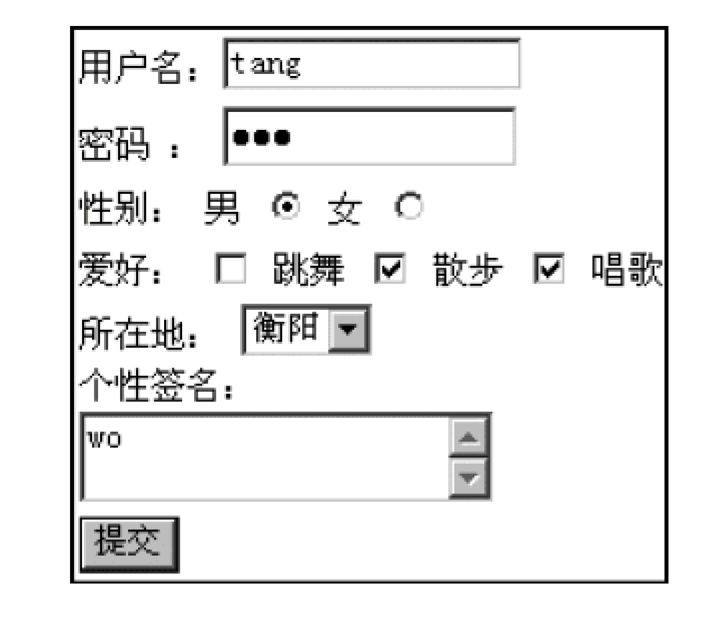


图 2-45 一个输入了数据的表单

```
<form action="login.php" method="post">
  <p>用户名：<input name="user" id="xm"
  type="text" size="15" /></p>
  <p>密码：<input name="pw" type="password"
  size="15" /></p>
  <p>性别：男<input type="radio" name="sex"
  value="1" />
     女<input type="radio" name="sex" value="2" /></p>
<p>爱好：<input name="fav1" type="checkbox"  value="1" />跳舞
           <input name="fav2" type="checkbox" value="2" />散步
           <input name="fav3" type="checkbox" value="3" />唱歌</p>
  <p>所在地：<select name="addr">
       <option value="1">长沙</option>
       <option value="2">湘潭</option>
       <option value="3">衡阳</option>
    </select></p>
  <p>个性签名： <br/><textarea name="sign"></textarea></p>
  <p><input type="submit" name="Submit" value="提交" /></p>
</form>
```

表单向服务器提交的内容总是 name/value 信息对，对于文本类输入框来说，value 的值是用户在文本框中输入的字符。对于选择框(单选框、复选框和列表菜单)来说，value 的值必须事先设定，只有某个选项被选中后它的 value 值才会生效。因此图 2-45 提交的数据是：

```
user=tang&pw=123&sex=1&fav2=2&fav3=3&addr=3&sign=wo&Submit=提交
```

提示：

（1）如果表单只有一个提交按钮，则可去掉它的 name 属性（如 name="Submit"），防止提交按钮的 name/value 属性对也一起发送给服务器，因为这些是多余的。

（2）<form>标记的 name 属性通常是为 JavaScript 调用该 form 元素提供方便的，没有其他用途。如果没有 JavaScript 调用该 form，则可省略其 name 属性。

2. 表单的三要素

一个最简单的表单必须具有以下三部分内容：

（1）<form>标记，没有它表单中的数据不知道提交到哪里去，并且不能确定这个表单的范围；

（2）至少有一个输入域（如 input 文本域或选择框等），这样才能收集到用户的信息，否则没有信息提交给服务器；

（3）提交按钮，没有它表单中的信息无法提交（当然，如果使用 Ajax 等高级技术提交表单，表单也可以不具有<form>标记和提交按钮，但本章不讨论这些）。

我们可查看百度首页中表单的源代码，这算是一个最简单的表单了，它的源代码如下（可见它具有上述的表单三要素，因此是一个完整的表单）：

```
<form name=f action=s>
    <input type=text name=wd id=kw size=42 maxlength=100>
    <input type=submit value=百度一下 id=sb>……
</form>
```

2.7.6　表单中的按钮

我们已经知道，在表单中可以用<input>标记创建按钮，只要设置它的 type 属性为 submit 就创建了一个提交按钮；设置 type 属性为 image 就创建了一个图像按钮，它们都可以用来提交表单；设置 type 属性为 reset 则是一个重置按钮，设置 type 属性为 button 就是一个普通按钮，它需要配合 JavaScript 脚本使其具有相应的功能，如表 2-10 所示。

表 2-10　用 input 标记创建按钮时的 type 属性类型设置

type 属性类型	功　能	作　　用
<input type="submit" />	提交按钮	提交表单信息
<input type="image" />	图像按钮	用图像做的提交按钮，也是提交表单信息
<input type="reset" />	重置按钮	将表单中的用户输入全部清空
<input type="button" />	普通按钮	需要配合 JavaScript 脚本使其具有相应的功能

但是，<input type="submit" />标记创建的按钮默认效果是没有图片的，而图像按钮虽然有图像但是不能添加文字。实际上，在 HTML 中有个<button>标记，它可以创建既带有图片又有文字的按钮，效果如图 2-46 所示。

登 录　注 册　登录

图 2-46　普通提交按钮、图像按钮与 button 标记创建的提交按钮比较

使用<button>标记创建按钮时的代码如下：

```
<button type="submit"><img src="check.png" align="absmiddle"/>登录
</button>
```

当然，还有一种思路是用 a 标记来模拟按钮，但那样就需要 CSS 和 JavaScript 的配合。通过 CSS 使 a 元素具有边框，再添加 JavaScript 脚本使其具有提交表单的功能。

2.7.7　表单的辅助标记

1. < label>标记

<label>标记用来为控件定义一个标签，它通过 for 属性绑定控件。如果表单控件的 id 属性值和 label 标记的 for 属性值相同，那么 label 标记就会和表单控件关联起来。通过在 Dreamweaver 中插入表单控件时选择“使用 for 属性附加标签标记”可快捷地插入 label 标记。例如：

```
<input type="radio" name="sex" value="radiobutton" id="male" />
        <label for="male">男</label><br />
<input type="radio" name="sex" value="radiobutton" id="female" />
        <label for="female">女</label>
```

添加带有 for 属性的<label>标记后，你会发现单击标签时就相当于单击了表单控件。

2. 字段集标记<fieldset>、<legend>

<fieldset>是字段集标记，它必须包含一个<legend>标记，表示是字段集的标题。如果表单中的控件较多，可以将逻辑上是一组的控件放在一个字段集内，这样显得有条理些。

2.7.8　HTML 5 新增的表单标记和属性

HTML 5 在表单方面做了很大的改进，包括：使用 type 属性增强表单，表单元素可以出现在 form 标记之外，input 元素新增了很多可用属性等。

1. input 标记的新增类型值

在 HTML 5 中，<input>标记在原有类型(type 属性值)的基础上，新增了许多新的类型成员，如表 2-11 所示。

表 2-11 <input>标记新增的类型

类型名称	type 属性	功能描述
网址输入框	<input type="url">	用来输入网址的文本框
Email 输入框	<input type="email">	用来输入 Email 地址的文本框
数字输入框	<input type="number">	输入数字的文本框，并可设置输入值的范围
范围滑动条	<input type="range">	可拖动滑动条，用于改变一定范围内的数字
日期选择框	<input type="date">	可选择日期的文本框
搜索输入框	<input type="search">	输入搜索关键字的文本框

其中，网址输入框与 Email 输入框虽然从外观上看与普通文本框相同，但是它会检测用户输入的文本是否是一个合法的网址或 Email 地址，从而不需要再使用 JavaScript 脚本来验证用户输入内容的有效性。

数字输入框示例代码如下，在 Google 浏览器中的外观如图 2-47(左)所示。

```
<input type="number" min="1960" max="1990" step="1" value="1980" />
```

相对于普通文本框，数字文本框会检验输入的内容是否为数字，并且可以设置数字的最小值(min)、最大值(max)和步进值(step)。当单击数字输入框右侧的上下箭头时，就会递增或递减当前值。

范围滑动条的示例代码如下，在 Chrome 浏览器中的外观如图 2-47(中)所示。

```
0<input type="range" min="0" max="20" value="10" />20
```

搜索输入框专门用于关键字查询，该类型输入框和普通文本框在功能和外观上没有太大区别，唯一的区别是：当用户在输入框中填写内容时，输入框右侧将会出现"×"按钮，单击该按钮，就会清空输入框中内容。示例代码如下，运行结果如图 2-47(右)所示。

```
<input name="keyword" type="search" />
```



图 2-47 数字输入框(左)、范围滑动条(中)和搜索输入框(右)的效果

日期选择框的示例代码如下，在 Google 浏览器中的外观如图 2-48 所示。

```
<input name="birth" type="date" value="2013-06-10" />
```

可见，日期选择框能够弹出日期界面供用户选择，如果对其设置 value 属性，则会显示该属性中的值作为默认日期。type 属性除了 date 外，将 type 属性设置为 time、month、week、datetime、datetime-local 均表示日期选择框，只不过此时能选择时间、月份、星期等值。

提示：如果浏览器不支持这些 HTML 5 中的 type



图 2-48 日期选择框

属性值，则会取 type 属性的默认值 text，从而将 input 元素解释为文本框。

2. input 标记新增的公共属性

在 HTML 5 中，input 标记新增了很多公共属性，如表 2-12 所示。除此之外，还新增了一些特有属性，如 range 类型中的 min、max、step 等。

表 2-12　input 标记新增的公共属性

属　　性	HTML 代码	功 能 说 明
autofocus	<input autofocus="true">	设置元素自动获得焦点
pattern	<input pattern="正则表达式">	使用正则表达式验证 input 元素的内容
placeholder	<input placeholder="默认内容">	设置文本输入框中的默认内容
required	<input required="true">	是否检测文本输入框中的内容为空
novalidate	<input novalidate="true">	是否验证文本输入框中的内容
autocomplete	<input autocomplete="on">	使 form 或 input 具有自动完成功能

（1）autofocus 属性：当 input 元素具有 autofocus 属性时，会使页面加载完成后，该元素自动获得焦点（即光标位于该输入框内）。

（2）pattern 属性：对于比较复杂的规则验证，如验证用户名“是否以字母开头，包含字符或数字和下划线，长度在 6～8 个字符之间”，则需要使用 pattern 属性设置正则表达式验证，例如，pattern="^[a-zA-Z]\w(5,7)$"。

（3）placeholder 属性：该属性可在文本框中放置一些提示文本（通常以灰色显示），当输入文本时，提示文本消失。示例代码如下，其效果类似于图 2-39。

```
<input name="keyword" type="search" placeholder="请输入关键字" />
```

（4）required 属性：该属性用来验证输入框的内容是否为空，如果为空，那么在表单提交时，会显示错误提示信息。

（5）novalidate 属性：该属性表示提交表单时不验证表单或输入框的内容，该属性适用于<form>以及以下类型的<input>标记：text、search、url、telephone、email、password、date pickers、range 以及 color。

（6）autocomplete 属性：该属性用来设置表单或输入框是否具有自动完成功能，其属性值是 on 或 off。开启自动完成功能后，当用户成功提交一次表单后，以后每次再提交表单时，都会在输入框下方出现以前输入过的内容供用户选择。

这些属性的功能过去一般都是用 JavaScript 脚本实现的，而现在用 HTML 5 属性实现后，可以大大减少对 JavaScript 代码的使用。

3. 新增的表单元素

在 HTML 5 中，除新增了 input 标记的类型外，还新增了许多新的表单元素，如 datalist、keygen、output 等。这些元素的加入，极大地丰富了表单数据的操作，优化了用

户体验。

1）datalist 元素

datalist 元素的功能是辅助表单中文本框的数据输入。datalist 元素本身是隐藏的，它需要与文本框的 list 属性绑定，只要将 list 属性值设置为 datalist 元素的 ID 属性即可。绑定成功后，用户在文本框输入内容时，datalist 元素将以列表的形式显示在文本框底部，提示输入的内容，与自动完成的功能类似。示例代码如下，运行效果如图 2-49 所示。

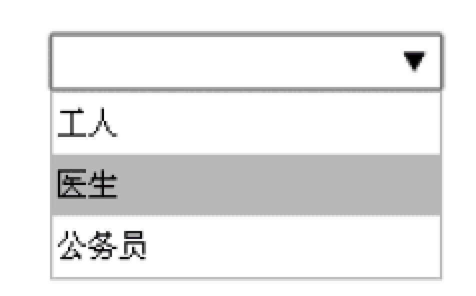


图 2-49 datalist 元素示例

```
<input type="text" id="zhiye" list="career" />
<datalist id="career">
    <option value="工人"></option>
    <option value="医生"></option>
    <option value="公务员"></option>
</datalist>
```

2）output 元素

output 元素的功能是在页面中显示各种不同类型表单元素的内容或运算后的结果，如输入框的值。output 元素需要配合 onFormInput 事件使用，在表单输入框中输入内容时，将触发该事件，从而可方便地获取到表单中各个元素的输入内容。下面是一个例子，当改变表单中两个文本框的值时，output 元素的值也随之改变。效果如图 2-50 所示。

```
<form oninput="x.value=parseInt(a.value)+parseInt(b.value)">
0<input type="range" id="a" value="50">100
    +<input type="number" id="b" value="50">
    =<output name="x" for="a b"></output>
</form>
```

3）keygen 元素

keygen 元素用于生成页面的密钥。如果在表单中添加该元素，那么当表单提交时，该元素将生成一对密钥：一个称为私钥，将保存在客户端；另一个称为公钥，将发送给服务器，由服务器进行保存，公钥可用于客户端证书的验证。

在表单中，插入一个 name 值为 userinfor 的<keygen>元素，代码如下：

```
<keygen name="userinfor" keytype="rsa" />
```

则会在页面中显示一个如图 2-51 所示的选择密钥位数的下拉列表框，当选择列表框中的密钥长度值后，提交表单，将根据所选择的密钥位数生成一对公私钥，并将公钥发送给服务器。

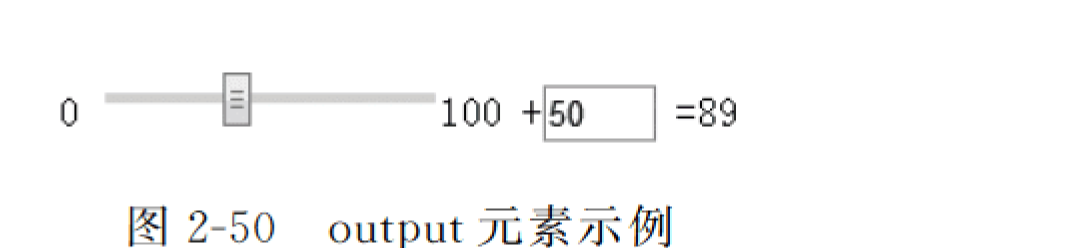


图 2-50 output 元素示例

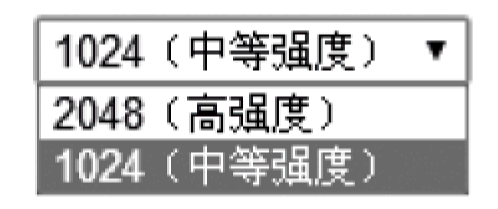


图 2-51 keygen 元素示例

目前，只有 Chrome、Firefox 和 Opera 浏览器支持该元素，因此，如果将 keygen 作为客户端安全保护的一种有效措施，还需要时间。

2.8 框架标记*

框架的作用是把浏览器的显示空间分割为几部分，每个部分可独立显示不同的网页。框架标记包括框架集<frameset>标记和框架<frame />标记，它们是成组出现的。

在互联网诞生初期，框架常用于网页排版，但现在用得很少了，一般只在网站后台管理程序中使用。通过一个框架集网页可使多个网页显示在一个浏览器窗口中。

2.8.1 <frameset>标记

窗口框架的分割有两种方式：一种是水平分割；另一种是垂直分割，在<frameset>标记中通过 cols 属性和 rows 属性来控制窗口的分割方式。框架标记的形式如下：

```
<frameset cols[或 rows]="各框架的大小或比例" border="像素值" bordercolor="颜色值" frameborder
="yes|no" framespacing="像素值">…</frameset>
```

如果要去掉框架的边框，可设置 frameborder="no"，framespacing 指框架和框架之间的距离，bordercolor 属性 IE 浏览器不支持。

1. 用 cols 或 rows 属性将分割窗口

cols 属性可以将一个框架集分割为左右若干列，每列就是一个框架，rows 属性用来将窗口分割成若干行个窗口。例如：

```
<frameset cols="n1,n2,…,*">
<frameset rows="30%,40%,*">
```

“n1,n2,…”表示每个子窗口的宽度，单位可以是像素或百分比。星号“*”表示分配给前面所有的窗口后剩余的宽度。在一个<frameset>标记中，不能同时设置 cols 属性和 rows 属性，否则 rows 属性将失效。

2. 框架的嵌套

通过框架的嵌套可实现对子窗口的分割，例如有时需要先将窗口水平分割，再将某个子窗口进行垂直分割，如图 2-52 所示，可用下面的代码实现。

```
<html>
<head><title>用框架分割窗体</title></head>
<frameset rows="30%,*">
  <frame src="2-8.html"/>
  <frameset cols="30%,*">
      <frame src="2-9.html"/><frame src="2-2hn.html"/>
  </frameset>
```

```
</frameset>
</html>
```

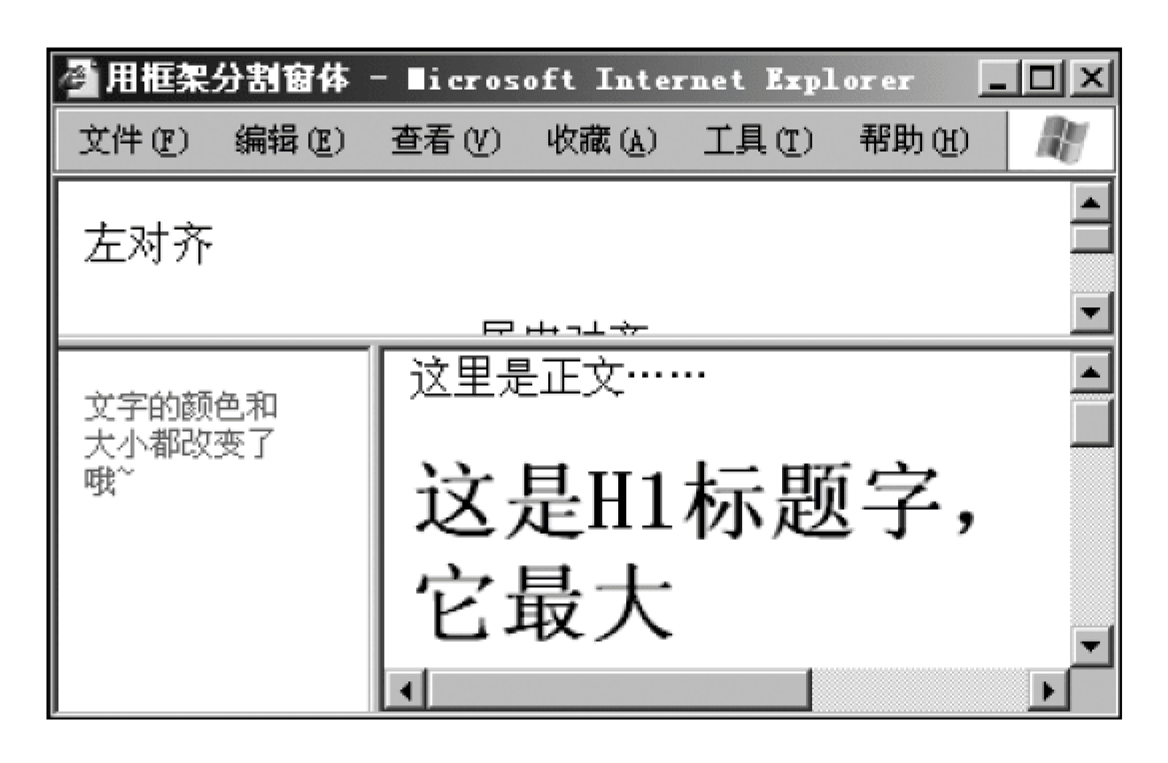


图 2-52 窗体的水平和垂直分割

需要注意的是，<frameset>标记和<body>标记是同级的，因此，不要将<frameset>标记写在<body>标记中，否则<frameset>标记将无法正常工作。

2.8.2 <frame/>标记

<frame/>标记是一个单标记，它的格式和常用属性如下：

```
<frame src="url" name="框架名" border="像素值" bordercolor="颜色值" frameborder="yes|no"
marginwidth="像素值" marginheight="像素值" scrolling="yes|no|auto" noresize="noresize" />
```

其中 scrolling 指定框架窗口是否允许出现滚动条，noresize 指定是否允许调整框架的大小。

1. 用 src 属性指定要显示的网页

框架的作用是显示网页，这是通过 src 属性来进行设置的。这个 src 属性和<img />中的 src 属性作用相似，都接文件的 URL。例如：

```
<frame src="demo/2-8.html"/>
```

2. 用 name 属性指定框架的名称

可以用 name 属性为框架指定名称，这样做的用途是：当其他框架中的链接要在指定的框架中打开时，可以设置其他框架中超链接的 target 属性值等于这个框架的 name 值。例如，在图 2-52 中，左边窗口中的链接都要求在右边窗口打开。那么可设置右边窗口的 name 值为 main，而左边窗口中所有链接的 target 属性值为 main。

例如，定义右边窗口 name 属性为 main：

```
<frame name="main"/>
```

左边窗口中的链接目标是 main：

```
<a href="add.htm" target="main">添加新闻</a>
```

这样 add.htm 会在框架名为 main 的窗口(右边窗口)中打开。

2.8.3 嵌入式框架标记<iframe>

框架标记只能对网页进行左右或上下分割,如果要让网页的中间某个矩形区域显示其他网页,则需要用到嵌入式框架标记,通过<iframe>可以很方便地在一个网页中显示另一个网页的内容,如图 2-53 网页中的天气预报就是通过 iframe 调用了另一个网页的内容。

下面是嵌入式框架的属性举例:

```
<iframe src="url" width="x" height="x" scrolling="[option]" frameborder="x" name="main"></iframe>
```

<iframe>标记中各个属性的含义如下:

(1) src——文件的 URL 路径。

(2) width、height——iframe 框架的宽和高。

(3) scrolling——当 src 指定的网页在区域中显示不完时,是否出现滚动条选项,如果设置为 no,则不出现滚动条;如果为 auto,则自动出现滚动条;如果为 yes,则显示。

(4) frameborder——iframe 边框的宽度,为了让框架与邻近内容相融合,常设置为 0。

(5) name——框架的名字,用来进行识别。例如:

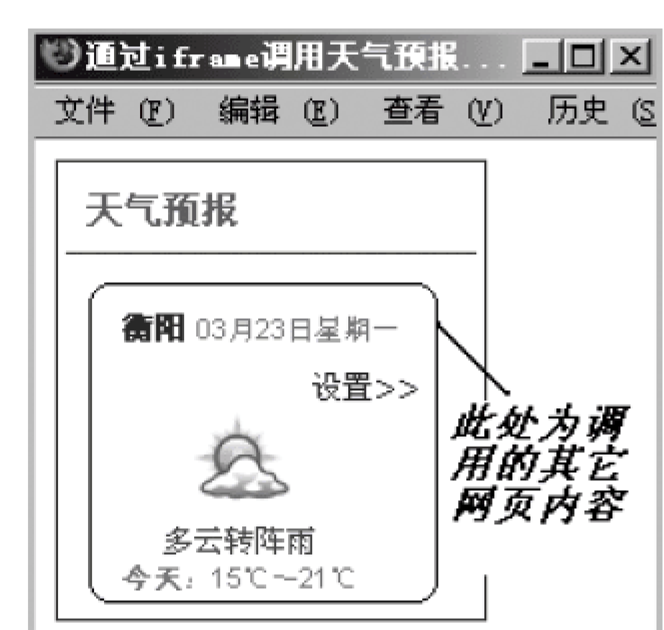


图 2-53 通过 iframe 调用天气预报网页

```
<iframe src="http://www.baidu.com" width="250" height="200" scrolling="auto" frameborder="0" name="main"></iframe>
```

嵌入式框架常用于将其他网页的内容导入到自己网页的某个区域,如把天气预报网站的天气导入到自己做的网页的某个区域显示。但某些木马或病毒程序利用 iframe 的这一特点,通过修改网站的网页源代码,在网页尾部添加 iframe 代码,导入其他带病毒的恶意网站的网页,并将 iframe 框架的宽和高都设置为 0,使 iframe 框架看不到。这样用户打开某网站网页的同时,就不知不觉打开了恶意网站的网页,从而感染病毒,这就是所谓的 iframe 挂木马的原理。不过可留意浏览器的状态栏看打开网页时是否提示正在打开某个可疑网站的网址而发现网页被挂木马。

提示:在 HTML 5 中已经不支持 frame 框架,但支持 iframe 框架。

2.9 头部标记*

如前所述,网页由 head 和 body 两个部分构成,在网页的 head 部分,除了 title 标记外,还有其他的几个标记,这些标记虽然不常用,但是需要有一定的了解。

1. <meta> 标记

meta 是元信息的意思,即描述信息的信息。<meta>标记提供网页文档的描述信息

等。如描述文档的编码方式、文档的摘要或关键字、文档的刷新，这些都不会显示在网页上。

<meta>标记可分为两类，如果它具有 name 属性，则表示它的作用是提供页面描述信息；如果它具有 http-equiv 属性，那么其作用就变成回应给浏览器一些有用的信息，以帮助正确和精确地显示网页内容。下面是几个例子。

（1）描述文档的编码方式，这可以防止浏览器显示乱码，例如“gb2312”表示简体中文。对于 XHTML 网页来说，这一项是必需的。在 Dreamweaver 中新建 XHTML 网页时，都自动会在网页头部有如下代码：

```
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
```

（2）描述摘要或关键字，网页的摘要，关键字是为了让搜索引擎能对网页内容的主题进行识别和分类。例如：

```
<meta name="Keywords" content="网页设计,学习" /><!--设置关键字-->
<meta name="Description" content="学习网页设计的网站" />  <!--设置摘要-->
```

（3）设置文档刷新。文档刷新可设置网页经过几秒钟后自动刷新或转到其他 URL。例如：

```
<meta http-equiv="refresh" content="30">  <!--过 30 秒后自动刷新-->
<meta http-equiv="refresh" content="5;? Url=index.htm"><!--5 秒后自动转到 index.htm -->
```

2. <link>、<style>和<script>标记

<link>标记用来链接外部 CSS 文件，<style>标记用来在网页头部嵌入 CSS 代码，而<script>标记用来链接或嵌入 JavaScript 代码。例如：

```
<link href="css/style.css" rel="stylesheet" type="text/css" />      <!--链接一个 css 文件-->
<style type="text/css">h1{font-size:12px;}</style>                  <!—嵌入 css 代码-->
<script src="js/jquery.js"></script>                                <!--链接一个外部 js 文件-->
<script>function msg() {alert("Hello")}</script>                    <!—嵌入 JavaScript 代码-->
```

习 题 2

1. HTML 中最大的标题元素是(　　)。
   A. <head>　　B. <title>　　C. <h1>　　D. <h6>
2. 下列哪种元素不能够相互嵌套使用?(　　)
   A. 表格　　B. 表单 form　　C. 列表　　D. div
3. 下述元素中(　　)都是表格中的元素。
   A. <table><head><th>　　B. <table><tr><td>
   C. <table><body><tr>　　D. <table><head><footer>
4. <title>标记中应该放在(　　)标记中。

A. <head> B. <table> C. <body> D. <div>

5. 下述(　　)表示表图像元素。

A. <img>image.gif</img> B.

C. D. <image src="image.gif " />

6. 要在新窗口打开一个链接指向的网页需用到(　　)。

A. href="_blank " B. name="_blank "

C. target="_blank " D. href="#blank "

7. align 属性的可取值不包括以下哪一项?(　　)

A. left B. center C. middle D. right

8. 下述哪一项表示表单控件元素中的下拉框元素?(　　)

A. <select> B. <input type="list">

C. <list> D. <input type="options">

9. 在下列的 HTML 中,哪个可以产生复选框?(　　)

A. <input type="check"> B. <checkbox>

C. <input type="checkbox"> D. <check>

10. 下列哪项表述是不正确的?(　　)

A. 单行文本框和多行文本框都是用相同的 HTML 标记创建的

B. 列表框和下拉列表框都是用相同的 HTML 标记创建的

C. 单行文本框和密码框都是用相同的 HTML 标记创建的

D. 使用图像按钮<input type="image">也能提交表单

11. colspan 是________标记的属性,cellpadding 是________标记的属性, target 是________标记或________标记的属性,<input>标记至少会具有________属性,<img>标记必须具有________属性,如果作为超链接,<a>标记必须具有________属性。

12. ________对象表示浏览器的 URL 地址,并可用于将浏览器转到某个网址。

13. 下面的表单元素代码都有错误,你能指出它们分别错在哪里吗?

① <input name="country" value="Your country here. " />

② <checkbox name="color" value="teal" />

③ <input type="password" value="pwd" />

④ <textarea name="essay" height="6" width="100">Your story.</textarea>

⑤ <select name="popsicle">

<option value="orange" /><option value="grape" /><option value="cherry" />

</select>

14. 画出下面 HTML 代码对应的表格:

```
<table width="466" height="127">
    <tr><td></td><td rowspan="2"></td></tr>
    <tr><td></td></tr></table>
```

15. 仿照图 2-36,设计一个用户注册的表单页面。

第3章 XHTML与Web标准

XHTML 是可扩展超文本标记语言（eXtensible HyperText Markup Language）的缩写，2000 年底，W3C 正式发布了 XHTML 1.0 版本。XHTML 1.0 是一种在 HTML 4.01 基础上做了少量优化和改进的新语言，是一种增强了的 HTML，是更严谨、更纯净的 HTML 版本。它是在 HTML 4.0 的基础上，为了适应 XML 而重新改造的 HTML，其特点是结合了部分 XML 的强大功能及大多数 HTML 的简单特性。

3.1 XHTML 与 HTML 的区别

从 HTML 到 XHTML 的过渡变化比较小，主要是为了适应 XML。其最大的变化在于文档必须是结构优良的，即所有标记必须闭合，也就是说，开始标记都要有相应的结束标记。另外，XHTML 中所有的标记必须小写。

3.1.1 文档类型的含义和选择

由于网页源文件存在不同的规范和版本，为了使浏览器能够兼容多种规范，在 XHTML 文档中，必须使用文档类型指令（DOCTYPE）来声明使用何种规范解释该文档。

目前，常用 HTML 或 XHTML 作为文档类型。而规范又规定，在 HTML 和 XHTML 中各自有不同的子类型，如严格类型（Strict）或过渡类型（Transitional）。其中，过滤类型兼容以前版本定义的，但在新版本中已经废弃的标记和属性；而严格类型则不兼容已经废弃的标记和属性。

建议读者使用 Dreamweaver 默认的 XHTML 1.0 Transitional（XHTML 1.0 过渡类型）作为网页的文档类型，这样既可以按照 XHTML 标准书写符合 Web 标准的网页代码，同时在一些特殊情况下还可以使用传统的做法。

在 Dreamweaver 中使用默认方式新建的网页文档在代码中的第一行都会有如下代码：

```
< !DOCTYPE html PUBLIC "- //W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1
- transitional.dtd">
```

这就是关于“文档类型”的声明，它告诉浏览器使用 XHTML 1.0 过渡规范来解释这个文档中的代码。其中 DTD 是文档类型定义（Document Type Definition）的英文首字母

缩写。

对 XHTML 文档类型的声明，有 Transitional、Strict 和 Frameset 三种子类型，Transitional 是过渡类型的 XHTML，表明兼容原来的 HTML 标记和属性；Strict 是严格型的应用方式，在这种形式下，不能使用 HTML 中任何样式表现的标记（如<font>）和属性（如 bgcolor）；Frameset 则是针对框架网页的应用方式，使用了框架的网页应使用这种类型。

注意：

(1) 文档类型声明必须写在 HTML 代码的第一行，且前面不能有任何空格，否则文档类型声明将失效；

(2) DOCTYPE 是用于定义文档类型的指令，但并不是一个标记，因此不需要封闭。

3.1.2　XHTML 与 HTML 的重要区别

尽管目前浏览器都兼容 HTML，但是为了使网页能够符合标准，读者应该尽量使用 XHTML 规范来编写代码，XHTML 的代码和 HTML 的代码有如下几个重要区别。

1. XHTML 文档必须在文档的第一行书写文档类型声明(DOCTYPE)

HTML 文档可以不写文档类型声明，但 XHTML 一定要有文档类型声明。

2. XHTML 文档可以定义命名空间

在 XHTML 文档中，<html>标记通常带有 xmlns 属性，例如：

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

xmlns 称为 XML 命名空间(XML NameSpace)，由于 XML 可以自定义标记，它需要用命名空间来唯一标识 XML 文档中的元素和实体的含义，通过特定 URL 关联命名空间文档，解决命名冲突，而 XHTML 是一种特殊的 XML，通过将 xmlns 修改为自定义命名空间文档的 URL，就可以自定义 XHTML 文档中的标记。例如，自定义一个<author>标记。但在一般情况下没必要修改命名空间，而且 xmlns 属性还可省略，这时将使用浏览器默认的命名空间。

3. XHTML 文档里必须具有 html、head、body、title 这些基本元素

对于 HTML 文档，即使代码里没有 html、head、body、title 这些基本元素，仍然是正确的，但 XHTML 要求一定要有这些基本元素，否则就不正确。

4. 在 XHTML 语言规范的基础上，对标记的书写有一些额外的要求

1) 标记名必须小写

HTML 中标记名既可大写又可小写，如<BODY><P>这是一个段落</P></BODY>，但在 XHTML 中则必须写成：

```
<body><p>这是一个段落</p></body>
```

2）属性名必须小写

例如，。

3）具有枚举类型的属性值必须小写

XHTML 并没有要求所有的属性值都必须小写，自定义的属性值可以大写，例如类名或 id 名的属性值可以使用大写字母，但枚举类型的属性值则必须要小写，枚举类型的值是指来自允许值列表中的值；例如，align 属性具有以下允许值：center、left 和 right。因此，下面的写法是符号 XHTML 标准的：

```
<div align="center" id="PageFooter">…</div>
```

4）属性值必须用双引号括起来

HTML 中，属性可以不必使用引号，例如，<img src=banner.jpg width=760 height=140>，而在 XHTML 中，必须严格写成：

```
<img src="banner.jpg" width="760" height="140" />
```

5）所有标记包括单标记都必须封闭

（1）这是指双标记必须要有结束标记，例如，<p>这是一个段落</p>。

（2）单标记也一定要用斜杠“/”封闭，例如，
、<hr />、等。

6）属性值必须使用完整形式

在 HTML 中，有些表单中元素的属性由于只有一个可选的属性值，通常就把这个属性值省略掉了，例如，<input checked>。

而在 XHTML 中，属性值在任何情况下都不能省略，例如

```
<input checked="checked" />
```

3.2 Web 标准

HTML 语言最开始是用来描述文档的结构的，如标题、段落等标记，后来因为人们还想用它控制文档的外观，HTML 便又增加了一些控制字体、对齐等方面的标记和属性，这样做的结果是 HTML 既可用来描述文档的结构，又能表示文档的外观，但是 HTML 描述文档表现的能力很弱，还造成了结构和表现混杂在一起，如果页面要改变外观，就必须重新编写 HTML，代码重用性低。

3.2.1 传统 HTML 的缺点

在 CSS 还没有被引入网页设计之前，传统的 HTML 语言要实现网页元素外观设计是十分麻烦的。例如，要在一个网页中把所有<h2>标记中的文字，设置为蓝色、黑体字显示。则需要在每一个 h2 标记中添加<font>标记，代码如下：

```
<h2><font color="#0000FF" face="黑体">h2标记 1</font></h2>
<p>CSS标记的正文内容 1</p>
```

```
<h2><font color="#0000FF" face="黑体">h2标记 2</font></h2>
<p>CSS标记的正文内容 2</p>
<h2><font color="#0000FF" face="黑体">h2标记 3</font></h2>
<p>CSS标记的正文内容 3</p>
<h2><font color="#0000FF" face="黑体">h2标记 4</font></h2>
<p>CSS标记的正文内容 4</p>
```

假设,网页中有 100 个 h2 标记,则需要重复添加 100 个 font 标记并设置属性,如果以后要将这 100 个标记的颜色修改为红色,则需要一个个地改,非常麻烦。

而使用 CSS 后,情况则完全不同,使用 CSS 实现上述功能的代码如下:

```
<html><head>
<style>
h2{                          /* 选中所有 h2标记 */
    font-family:"黑体";
    color:blue;              /* 设置字体颜色 */
}
</style>
</head>
<body>
    <h2>h2标记 1</h2><p>CSS标记的正文内容 1</p>
    <h2>h2标记 2</h2><p>CSS标记的正文内容 2</p>
    <h2>h2标记 3</h2><p>CSS标记的正文内容 3</p>
    <h2>h2标记 4</h2><p>CSS标记的正文内容 4</p>
</body></html>
```

这样,只要修改上述 CSS 代码中的字体颜色属性值 blue,就可以改变页面中所有 h2 标记的颜色。并且,CSS 还能统一设置网站中所有页面字体的风格。

3.2.2　Web 标准的含义

为了让网页的结构和表现能够分离,W3C 提出了 Web 标准,即网页由结构、表现和行为组成。用 HTML 的新版本 XHTML 描述文档的结构,用 CSS 控制文档的表现,因此 XHTML 和 CSS 就是内容和形式的关系,由 XHTML 确定网页的内容,而通过 CSS 来决定页面的表现形式。

Web 标准是指网页由结构(Structure)、表现(Presentation)和行为(Behavior)组成,为了理解 Web 标准,就需要明确下面几个概念。

(1) 内容:内容就是页面实际要传达的真正信息,包含文本或者图片等。注意这里强调的“真正”,是指纯粹的数据信息本身。例如:

```
天仙子 (1)宋.张先 沙上并禽池上暝,云破月来花弄影。重重帘幕密遮灯,风不定,人初静,明日落红
应满径。作者介绍张先(990—1078)字子野,乌程(今浙江湖州)人。天圣八年(1030)进士。官至尚书
都官郎中。与柳永齐名,号称"张三影"。
```

(2) 结构:可以看到上面的文本信息本身已经完整。但是混乱一团,难以阅读和理解,我们必须给它格式化一下。把它分成标题、作者、章、节、段落和列表等。例如:

标题 天仙子 (1)
作者 宋.张先
正文
沙上并禽池上暝,云破月来花弄影。
重重帘幕密遮灯,风不定,人初静,
明日落红应满径。
节 1作者介绍
张先(990—1078)字子野,乌程(今浙江湖州)人。天圣八年(1030)进士。官至尚书都官郎中。与柳永齐名,号称"张三影"。

(3) 表现:上面的文档虽然定义了结构,但是内容还是原来的样式没有改变,例如标题字体没有变大,正文的颜色也没有变化,没有背景,没有修饰。所有这些用来改变内容外观的东西,我们称之为“表现”。下面对它增加这些修饰内容外观的东西,修饰后的效果如图 3-1 所示。

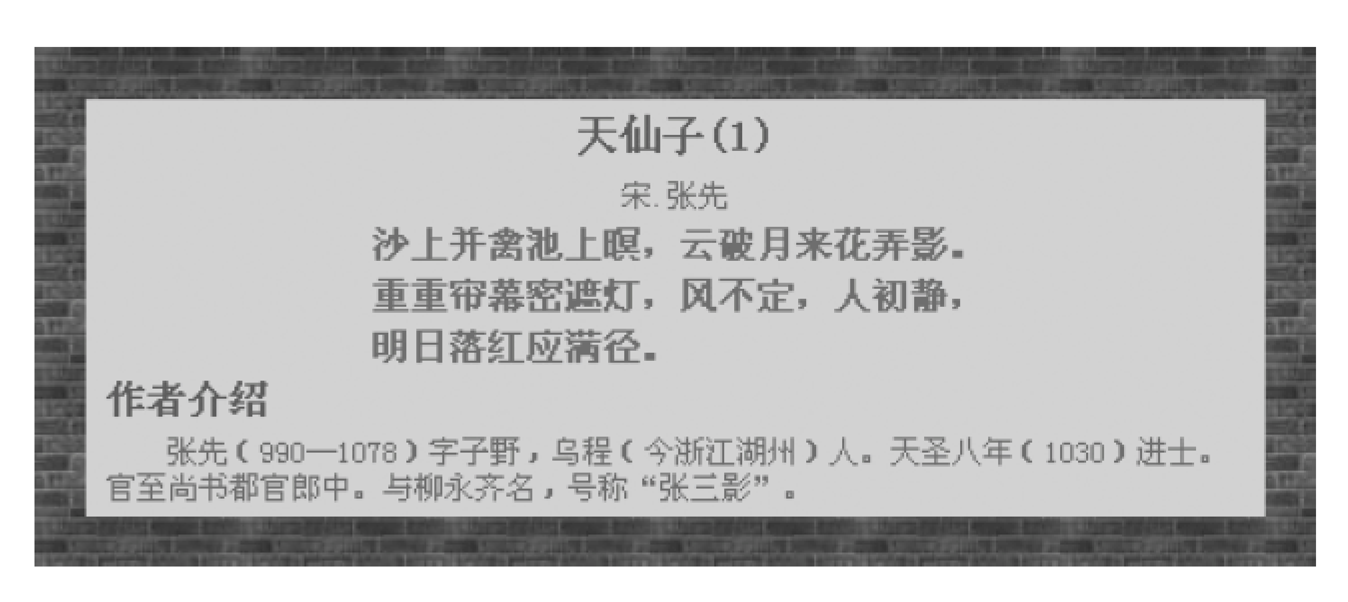


图 3-1　文档添加了“表现”后的效果

很明显,可以看到我们对文档加了两种背景,将标题字体变大并居中,将小标题加粗并变成红色,等等。所有这些,都是“表现”的作用。它使内容看上去漂亮、可爱多了!形象一点的比喻:内容是模特,结构标明头和四肢等各个部位,表现则是服装,将模特打扮得漂漂亮亮。

(4) 行为:就是对内容的交互及操作效果。例如,使用 JavaScript 可以响应鼠标的单击和移动,可以判断一些表单提交,使我们的操作能够和网页进行交互。

所以说,网页就是由这四层信息构成的一个共同体,这四层的作用如图 3-2 所示。

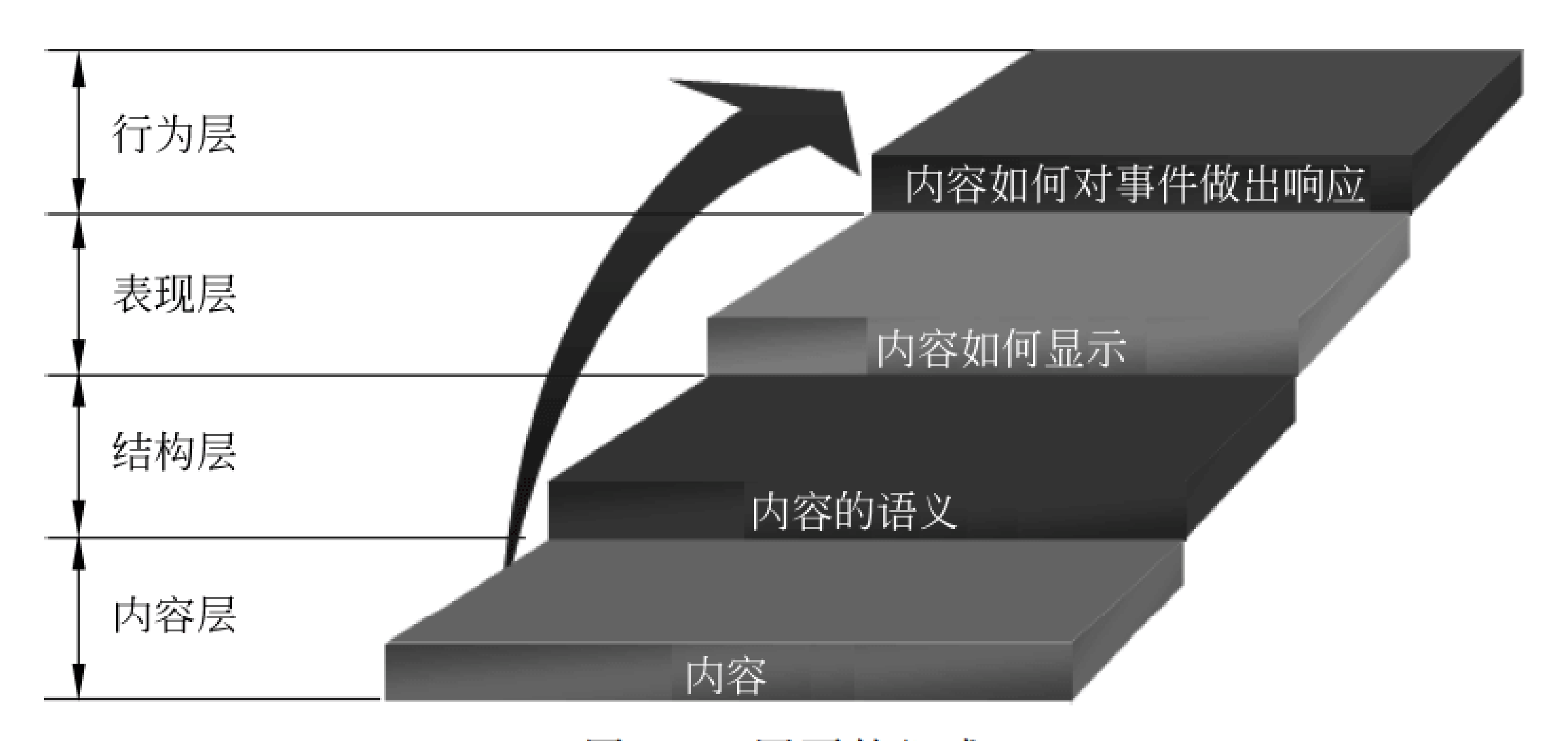


图 3-2　网页的组成

在 Web 标准中,结构标准语言是指 XML 和 XHTML,表现标准语言是指 CSS(Cascading Style Sheets,层叠样式表),行为标准语言主要指 JavaScript。但是实际上 XHTML 语言也有很弱的描述表现的能力,而 CSS 也有一定的响应行为的能力(如 hover 伪类),而 JavaScript 是专门为网页添加行为的。所以这三种语言对应的功能总体来说如图 3-3 所示,并且这三种语言是相互关联密切配合的,它们的关系如图 3-4 所示。

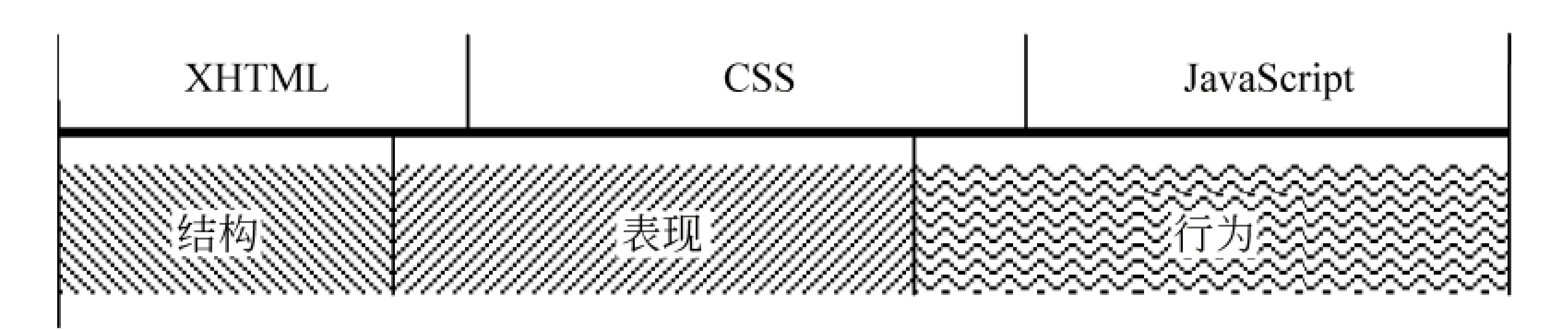


图 3-3 网页的组成项及实现它们的语言

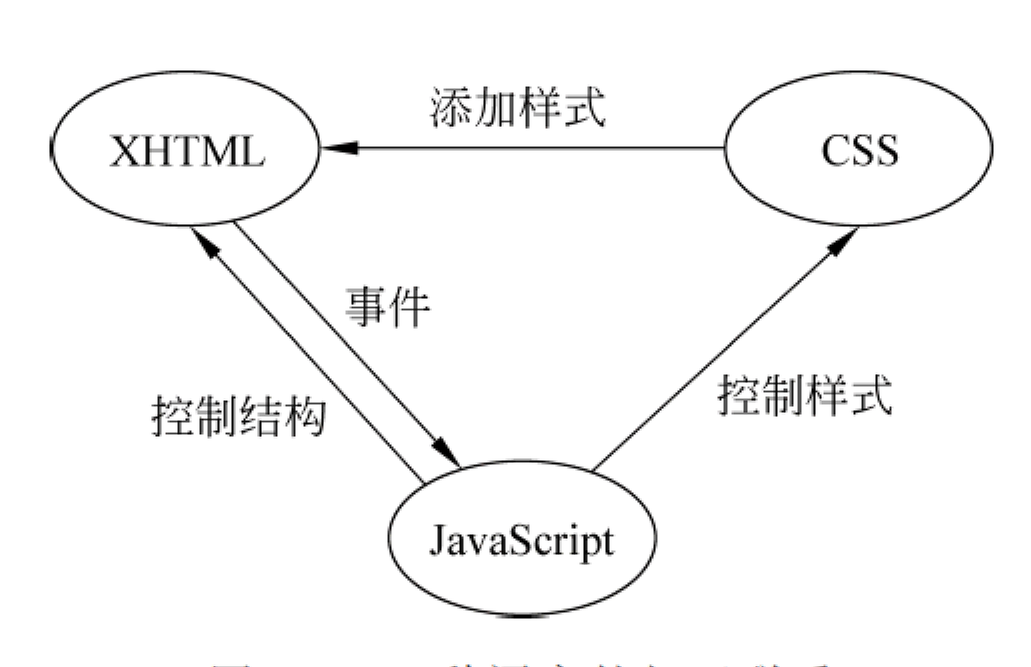


图 3-4 三种语言的相互联系

3.2.3 Web 标准的优势

Web 标准的核心思想就是“结构”和“表现”相分离,让 HTML 和 CSS 各司其职,这样做的好处有以下几点:

(1) 由于使用 CSS 代码统一设置元素样式,可以大量减少 HTML 代码的使用,从而减小网页文件的体积,使页面载入、显示速度更快,并降低网站流量费用;

(2) 使用 CSS 统一设置大量 HTML 元素的样式后,修改网页时更有效率而且代价更低;

(3) 在 Web 标准中推荐使用有语义的 HTML 元素定义内容,如使用 h1 标记定义标题,这样搜索引擎就能更好地理解网页中的内容,对搜索引擎更加友好,有利于搜索引擎优化(Search Engine Optimization,SEO),从而提高网站在搜索引擎中的排名;

(4) 使网站对浏览器更具亲和力,遵循 Web 标准设计的网页由于具有良好的文档结构,使不能有效解析 HTML 文档的盲人设备或手持设备也能理解网页代码内容。

大体来看,Web 标准是从 2004 年开始在我国逐渐风靡起来的,在这之前由于 IE 5.5 以下版本浏览器对 CSS 的支持很不好,人们只能更多地使用 HTML,想尽办法使 HTML 同时承担着“结构”和“表现”的双重任务。直到 Windows XP 普及,其内置的 IE 6 对 CSS 的支持显著改善,使我国设计师开始重视 CSS,并逐渐遵循 Web 标准来设计网页了。

3.3 HTML 元素的概念

HTML 文档是由各种 HTML 元素组成的,网页中文字、图像、链接等所有的内容都是以元素的形式定义在 HTML 代码中的,因此元素是构成 HTML 文档的基本部件。元素是用标记来表现的,一般起始标记表示元素的开始,结束标记表示元素的结束。把 HTML 标记(如<p>…</p>)和标记之间的内容组合称为元素。

HTML 元素可分为“有内容的元素”和“空元素”两种。“有内容的元素”是由起始标记、结束标记和两者之间的内容组成,其中元素内容既可以是文字内容,也可以是其他元素。例如,一个 b 元素(这是粗体字),它的元素内容是文字“这是粗体字”;而 html 元素的内容是另两个元素(head 元素和 body 元素)。“空元素”则只有起始标记而没有结束标记和元素内容。例如,
元素就是空元素,可见“空元素”对应单标记。

标记相同而标记中的内容不同应视为不同的元素,同一网页中标记和标记的内容都相同的元素如果出现两次也应视为两个不同的元素,因为浏览器在解释 HTML 中每个元素时都会为它自动分配一个内部 ID,不存在两个元素的 ID 也相同的情况。

例 3.1 在如下代码中,body 标记内共有多少个元素?

```
<html><body>
    <a href="box.html"><img src="cup.gif" border="0" align="left" /></a>
    <p>图片的说明内容</p><hr/>
    <p>图片的说明内容</p>
</body></html>
```

答案:5 个,即 1 个 a 元素、1 个 img 元素、2 个 p 元素和 1 个 hr 元素。

3.3.1 行内元素和块级元素

HTML 元素还可以按另一种方式分为“行内元素”和“块级元素”。下面是一段 HTML 代码,它的显示效果如图 3-5 所示,请注意元素中的内容在浏览器中是如何排列的。

```
<html><body>
    <h2>web 标准</h2><a href="#">w3c 主页</a>
    <img src="arrow.gif" width="16" height="16" /><b>结构</b>
    <font>表现</font><span>行为</span>
    <p>结构标准语言 XHTML</p><ul><li>表现标准语言 CSS</li></ul>
    <div>行为标准语言 JavaScript</div>
</body></html>
```

从图 3-5 中可以看到,h2、p、div 这些元素中的内容会占满一整行,而 a、img、span 这些元素在一行内从左到右排列,它们占据的宽度刚好能容纳元素中内容的最小宽度。根据元素是否会占据一整行,可以把 HTML 元素分为行内元素和块级元素。

行内(inline)元素是指元素与元素之间从左到右并排排列，只有当浏览器窗口容纳不下才会转到下一行，块级(block)元素是指每个元素占据浏览器一整行位置，块级元素与块级元素之间自动换行，从上到下排列。块级元素内部可包含行内元素或块级元素，行内元素内部可包含行内元素，但不得包含块级元素。另外，块级元素<p>元素内部也不能包含其他的块级元素。



图 3-5　行内元素和块级元素

3.3.2　div 和 span 标记

<div>和<span>是不含有语义的标记，用来在标记中放置任何网页元素(如文本、图像等)。就像一个容器一样，当把内容放入后，内容的外观不会发生任何改变，这样有利于内容和表现分离。应用容器标记的主要作用是通过引入 CSS 属性对容器内的内容进行设置。div 和 span 唯一的区别是：div 是块级元素，span 是行内元素。下面是一段示例代码，显示效果如图 3-6 所示。

```
<html><body>
    <div>div 元素 1</div>      <div>div 元素 1</div>
    <span>span 元素 1</span><span>span 元素 2</span>
</body></html>
```

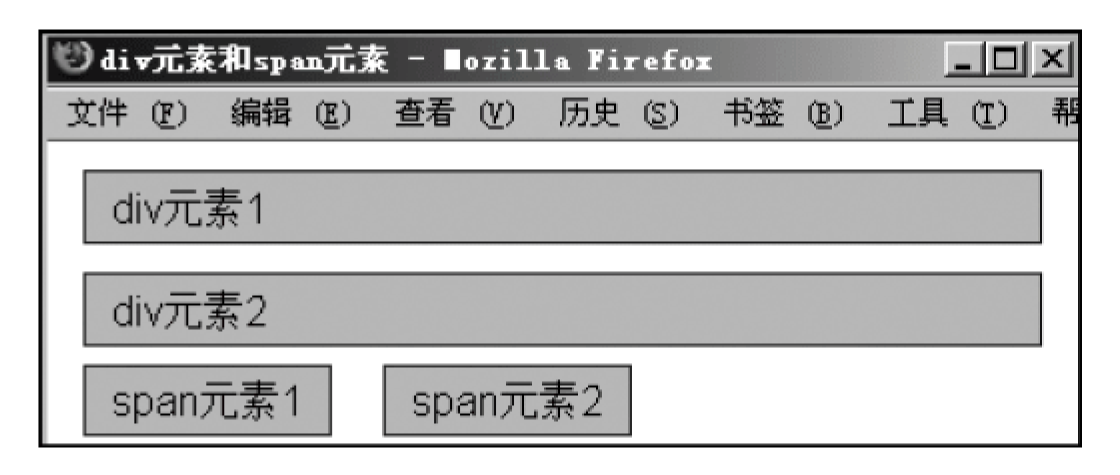


图 3-6　div 元素和 span 元素的区别(利用 CSS 为每个元素添加了背景和边框属性)

可以看出 div 元素作为块级元素会占满整个一行，两个元素间上下排列；而 span 元素的宽度不会自动伸展，以能包含它的内容的最小宽度为准，两个元素之间从左到右依次排列。

需要注意的是，div 元素并不对应于“层”的概念，过去说的层是指通过 CSS 设置成了绝对定位的 div 元素，但实际上也可以对其他任何标记(如<p>)的元素设置成绝对定位，此时其他元素也成了“层”。因此层并不对应于任何 HTML 标记，所以 Dreamweaver CS3 去掉了层这一概念，将这些设置成了绝对定位的元素统称为 AP(Absolute Position)元素。

3.4　HTML 5 简介

HTML 5 是 HTML 语言的最新版本，其前身是由网页超文本应用技术工作小组 WHATWG(Web Hypertext Application Technology Working Group)于 2004 年提出的

Web Applications 1.0。在2007年被W3C接纳，并成立了新的HTML工作团队。HTML 5的正式版本于2010年9月发布。

HTML 5已经被IE 10＋、Chrome 18＋、Firefox 4、Safari 5等浏览器支持，对于不支持HTML 5的旧版浏览器，HTML 5也能保证旧版浏览器能够安全地忽略掉HTML 5代码，力图让不同的浏览器即使在发生语法错误时也能返回相似的显示结果。

3.4.1 HTML 5新增的标记

与HTML 4.01相比，HTML 5提供了一些新的标记和属性，这些新增的标记主要可分为：

(1) 文档结构标记，例如<nav>(网站导航条区域)和<footer>(网站底部区域)等。这些标记将有利于搜索引擎的索引整理，同时更好地帮助小屏幕装置和视障人士使用。

(2) 媒体元素标记，如<audio>和<video>标记。

(3) 表单标记等。具体如表3-1所示。

表3-1 HTML 5新增的标记

标记名	格 式	用 法
<video>	<video src="" width="" …> …</video>	插入视频
<audio>	<video src="" width="" …> … </video>	插入音频
<canvas>	<canvas id="" width="" …>… </canvas>	画布标记，用来绘制图形
<command>	<command type=""> …</command>	定义命令按钮
<datalist>	<datalist id=" ">…</datalist>	定义输入框的附带下拉列表
<meter>	<meter value="" min="" max="" low="" high="">…</meter>	定义数值条
<progress>	<progress value="" max="">…</progress>	定义进度条
<time>	<time datetime=""></time>	定义日期或时间
<summary>	<summary>…</summary>	定义元素的摘要
<details>	<details><summary>…</summary>…</details>	定义元素的细节，常与summary标记配合
<figure>	<figure></figure>	定义媒介内容的分组，以及它们的标题
<mark>	… <mark>突出的文本</mark> …	给文本加背景色以突出显示
<ruby>	<ruby>ruby注释 <rt>解释</rt></ruby>	定义ruby语言的注释
<rt>	同上	定义ruby注释的解释
<wbr>	XML<wbr>Http<wbr>Request	页面宽度不足时，一个单词内字母换行的位置

下面是几个HTML 5标记的使用示例。

1. <meter>与<progress>标记

<meter>与<progress>属于状态交互元素，其示例代码如下，运行效果如图 3-7 所示。其中，value 属性用于设置元素展示的实际值，默认为 0；min 和 max 用于设置元素展示的最小值和最大值，low 和 high 用于设置元素展示的最低值和最高值。其范围应该在 min 和 max 值的范围以内。

```
<p>速度：<meter value="120" min="0" max="220" low="0" high="160">
    120</meter>km</p>
<p>剩余油量：<progress value="30" max="100">30/100</progress></p>
```

2. <details>与<summary>标记

<details>标记初始时只会显示其中<summary>标记的内容，当用户单击<summary>标记时，会展开显示<details>标记的所有内容。示例代码如下，网页载入时只显示“衡阳师范学院”，当用户单击“衡阳师范学院”时，其运行效果如图 3-8 所示。

```
<details>
    <summary>衡阳师范学院</summary>
        <p>湖南省直属的一所普通全日制公办本科院校</p>
</details>
```

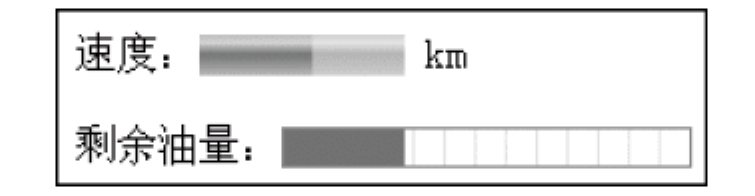


图 3-7　<meter>与<progress>标记示例

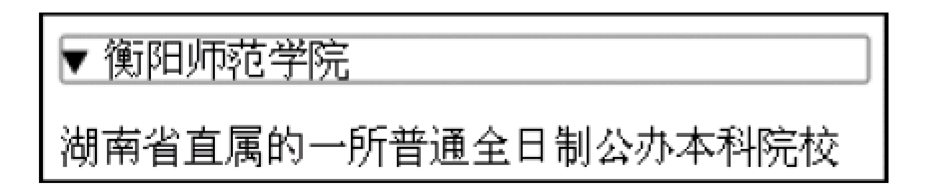


图 3-8　<details>与<summary>标记示例

提示：在 HTML 5 中，已经取消了一些过时的 HTML 4 标记。这主要包括：

(1) 字体标记，如<font>、<b>、<center>、<marquee>等，它们已经被 CSS 取代；

(2) Java 小程序嵌入标记<applet>；

(3) 框架标记<frameset>、<frame>等。

3.4.2　HTML 5 语法的改进

1. 文档类型声明的改进

在 HTML 5 文档中声明文档类型的(DOCTYPE)的代码已经变得非常简洁，代码如下：

```
<!DOCTYPE html>
```

可见，在 HTML 5 中已不再有 3 种不同文档类型(过渡型、严格型和框架型)的区别。

2. 指定字符编码

如果想指定文档使用的字符编码，HTML 5 仍然使用 meta 属性，但代码已经简化

如下：

```
<meta charset="utf-8">
```

3. 属性书写的简化

HTML 5 对标记和属性的写法又回归到了简化的风格，这包括：属性如果只有唯一值（如 checked），则可省略属性值，属性值两边的引号也可省略，例如，下面的写法都是正确的：

```
<input type="text" name="pwd" required>
<img src=foo alt=bar>
<p class=foo>Hello world</p>
```

4. 超链接可以包含块级元素

在过去，要想给很多块级元素添加超链接，只能在每个块级元素内嵌入<a>标记，在 HTML 5 中，只要简单地把所有内容都写在一个链接元素中就可以了。示例代码如下：

```
<a href="#">
    <h2>标题文本</h2>
    <p>段落文本</p>
</a>
```

习 题 3

1. 在 XHTML 中必须声明文档类型，以便于浏览器知道当前浏览的文档的类型，声明文档类型需使用指令________。

2. Web 标准主要由一系列规范组成，目前的 Web 标准主要由三大部分组成________、________、________。

3. 所有 XHTML 标记必须________和________。

4. 在 XHTML 中，有 3 种文档类型声明 DTD，分别是____________、____________和________。

第 4 章 CSS

CSS(Cascading Styles Sheets,层叠样式表)是用于控制网页样式并允许将样式信息与网页内容分离的一种标记性语言。HTML 和 CSS 就是“内容”和“形式”的关系,由 HTML 组织网页内容的结构,而通过 CSS 来决定页面的表现形式。CSS 和 XHTML 都是由 W3C 负责组织和制定的。

由于 HTML 的主要功能是描述网页的结构,所以控制网页外观的能力很差,如无法精确调整文字大小、行距等,而且不能对多个网页元素进行统一的样式设置,只能一个一个元素地设置。使用 CSS 可实现对网页的外观和排版进行更灵活的控制,使网页更美观。

4.1 CSS 基础

CSS 样式表是由一系列样式规则组成的,浏览器将这些规则应用到相应的元素上,CSS 语言实际上是一种描述 HTML 元素外观(样式)的语言,下面是一条 CSS 样式规则和描述一个人的特征的规则的对比。

```
h1 {                              ⟹        关羽 {
    color: red;                                 身高: 185cm;
    font-size: 25px;                            体重: 95kg;
  }                                           }
```

4.1.1 CSS 的语法

一条 CSS 样式规则由选择器(Selector)和声明(Declarations)组成,如图 4-1 所示。选择器是为了选中网页中某些元素的,也就是告诉浏览器,这段 CSS 样式规则将应用到哪组元素上。

选择器用来定义 CSS 规则的作用对象,它可以是一个标记名,表示将网页中所有该标记的元素全部选中。如图 4-1 中的 h1 选择器就是一个标记选择器,它将网页中所有 <h1> 标记的元素全部选中,而声明则用于定义元素样式。介于花括号{ }之间的所有内容都是声明,声明又分为属性(Property)和值(value),图 4-1 中的示例为所有 <h1> 标记

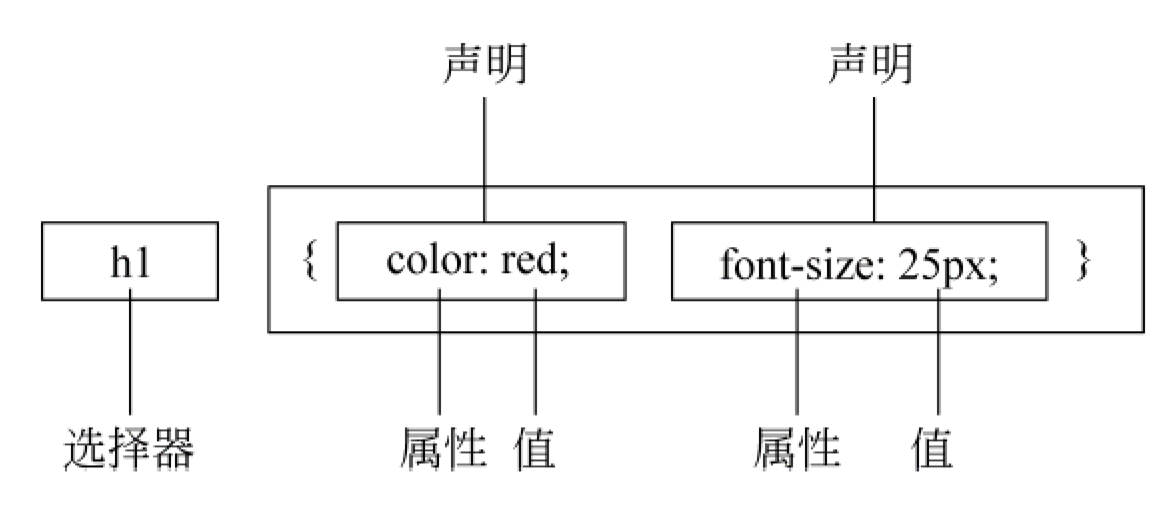


图 4-1 CSS 样式规则的组成(标记选择器)

的元素定义了 2 个属性,使该网页中所有 h1 元素的文本都将是红色并且是 25 像素大小。

属性是 CSS 样式控制的核心,对于每个 HTML 元素,CSS 都提供了丰富的样式属性,如颜色、大小、背景、盒子、定位等。表 4-1 列出了一些最常用的 CSS 属性。

表 4-1 最常用的 CSS 属性

CSS 属性	含 义	举 例
font-size	字体大小	font-size:14px;
color	字体颜色(仅能设置字体的颜色)	color:red;
line-height	行高	line-height:160%;
text-decoration	文本修饰(如增删下划线)	text-decoration:none;
text-indent	文本缩进	text-indent:2em;
background-color	背景颜色	background-color:#ffeeaa;

CSS 的属性和值之间用冒号隔开(注意 CSS 属性和值的写法与 HTML 属性的区别)。如果要设置多个属性和值,可以书写多条声明,每条声明之间用分号隔开。

对于属性值的书写,有以下规则:

- 如果属性的某个值不是一个单词,则值要用引号引起来,如

```
p {font-family: "sans serif"}
```

- 如果一个属性有多个值,则每个值之间要用空格隔开,如

```
a {padding: 6px 4px 3px}
```

- 如果要为某个属性设置多个候选值,则每个值之间用逗号隔开,如

```
p {font-family: "Times New Roman", Times, serif}
```

4.1.2 在 HTML 中引入 CSS 的方法

HTML 和 CSS 是两种作用不同的语言,它们同时对一个网页产生作用,必须通过一些方法将 CSS 与 HTML 挂接在一起,才能正常工作。

在 HTML 中,引入 CSS 的方法有行内式、嵌入式、导入式和链接式 4 种。

1. 行内式

所有 HTML 标记都有一个通用的属性 style,行内式就是将元素的 CSS 规则作为

style 属性的属性值写在元素的标记内，例如：

```
<td style="color: red; text-decoration: underline" width="92% ">
```

行内式引入的优点是：由于 CSS 规则就写在标记内，其作用对象就是该元素，所以无须书写 CSS 的选择器。有时需要做测试或对个别元素设置 CSS 属性，这时可以使用这种方式，只需要书写属性和值，但它没有体现出 CSS 统一设置许多元素样式的优势。

2. 嵌入式

嵌入式将页面中各种元素的 CSS 样式设置集中写在<style>和</style>之间，<style>标记是专用于引入嵌入式 CSS 的一个 html 标记，它只能放置在文档头部，即<style>…</style>只能放置在文档的<head>和</head>之间。例如：

```
<head>
<style type="text/css">      <!--该标记用来嵌入 css 代码-->
    h1{
        color: red;
        font-size: 25px;      }
</style>
</head>
```

为单一的网页设置样式，嵌入式方式很方便且最常用。但是对于一个包含很多网页的网站来说，如果每个网页都以嵌入式的方式设置各自的样式，不仅麻烦，冗余代码多，而且网站中各个页面的风格不好统一。因此，对于一个网站来说，通常都是编写独立的 CSS 文件，使用以下两种方式之一，引入到网站的所有 HTML 网页文档中。

3. 链接式和导入式

当 CSS 样式需要应用于很多页面时，外部样式表(外部 CSS 文件)将是理想的选择。所谓外部样式表，就是将 CSS 规则写入到一个单独的文本文件中，并将该文件的后缀名命名为.css。然后使用链接式或导入式的方法将外部 CSS 文件引入到 HTML 文件中，其优点是可以让很多个网页共享一个 CSS 文件设置的样式。

在学习 CSS 或制作单个网页时，为了方便可采取行内式或嵌入式方法引入 CSS，但若要制作网站，则主要应采用链接式引入外部 CSS 文件，以便使网站内的所有网页风格统一。而且在使用外部样式表的情况下，可以通过改变一个外部 CSS 文件来改变整个网站所有页面的外观。下面介绍引入外部 CSS 文件的方法。

链接式是在网页头部使用 HTML 标记<link>引入外部 CSS 文件，语法如下：

```
<link href="style1.css" rel="stylesheet" type="text/css" />
```

其中 style1.css 的文件内容如下(注意该文件中不需要书写<style>标记)：

```
h1 {
  color: red;
  font-size: 25px;
  }
```

而导入式是通过 CSS 规则中的@import 指令来导入外部 CSS 文件，语法如下：

```
<style type="text/css">
    @import url("style1.css");
</style>
```

此外，这两种方式的显示效果也略有不同。使用链接式时，会在装载页面主体部分之前装载 CSS 文件，这样显示出来的网页从一开始就是带有样式效果的；而使用导入式时，要在整个页面装载完之后再装载 CSS 文件，如果页面文件比较大，则开始装载时会显示无样式的页面。从浏览者的感受来说，这是使用导入式的一个缺陷。

4.1.3 选择器的分类

选择器就是为了选中文档中要应用样式的那些元素，为了能够灵活选中文档中的某类或某些元素，CSS 定义了很多种选择器。其中，基本的 CSS 选择器包括标记选择器、类选择器、ID 选择器和伪类选择器 4 种。

1. 标记选择器

标记是元素的固有特征，标记选择器用来声明哪种标记采用哪种 CSS 样式。因此，每一个 HTML 标记的名称都可以作为相应的标记选择器的名称，标记选择器形式如图 4-1 所示，它将属于该标记的所有元素全部选中。例如：

```
<style type="text/css">
p{                              /* 标记选择器 */
    color:blue;
    font-size:18px;     }
</style>
    <p>选择器之标记选择器 1</p>
    <p>选择器之标记选择器 2</p>
    <p>选择器之标记选择器 3</p>
    <h3>h3 则不适用</h3>
```

以上所有 3 个 p 元素都会应用 p 标记选择器定义的样式，显示为“蓝色，18 像素大”，而 h3 元素则不会受到影响。

提示： 本书对代码采用了简略写法，书中 CSS 代码主要采用嵌入式方式引入 HTML 文档中。因此，读者只要将代码中<style>…</style>部分放置在文档的<head>和</head>之间，将其他 HTML 代码放置在<body>和</body>之间，就能还原成可运行的原始代码。

2. 类选择器

标记选择器一旦声明，那么页面中所有该标记的元素都会产生相应的变化。例如，当声明<p>标记为红色时，页面中所有的 p 元素都将显示为红色。但是如果希望其中某一些 p 元素不是红色，而是蓝色，就需要将这些 p 元素自定义为一类，用类选择器来选中它

们;或者希望不同标记的元素属于同一类,应用同一样式,如希望某些 p 元素和 h3 元素都是蓝色,则可以将这些不同标记的元素定义为同一类。也就是说,标记选择器根据元素的固有特征(标记名)分类,好比人可以根据固有特征"肤色"分为黄种人、黑种人和白种人,而类选择器是人为地对元素分类,比如人又可以分为教师、医生、公务员等这些社会自定义的类别。

要应用类选择器,首先应给相应的 HTML 元素添加一个通用 HTML 属性 class,只要对不同的元素定义相同的类名,那么这些元素将被划分成同一类,例如:

```
<h3 class="test">将该元素划入 test 类</h3>
<p class="test">将该元素划入 test 类</p>
```

然后再根据类名定义类选择器来选中该类元素,类选择器以半角". "开头,格式如下:

```
.test{ color: red; font-size:20px; }
```

下面的代码中定义了 2 个类选择器,并为 4 个 HTML 元素应用了类选择器的样式。

```
<style type="text/css">
.one{                    /* 类选择器 .one */
    color: red;          /* 字体颜色红色 */
}
.two{                    /* 类选择器 .two */
    font-size:20px;      /* 文字大小 20 像素 */
}
</style>
    <p>选择器之标记选择器 1</p>
    <p class="one">应用第一种 class 选择器样式</p>
    <p class="two">应用第二种 class 选择器样式</p>
    <p class="one two">同时应两种 class 选择器样式</p>
    <h3 class="two">h3 同样适用</h3>
```

其中第 3 个 p 元素和 h3 元素被定义成了同一类,而第 4 行通过 class="one two"将同时应用两种类选择器的样式,得到红色 20 像素的大字体。对一个元素定义多个类名是允许的,就好像一个人可能既属于教师又属于作家一样。第一行的 p 元素因未定义类别名则不受影响,仅作为对比。

3. ID 选择器

ID 选择器的使用方法与类选择器基本相同。不同之处在于一个 ID 选择器只能应用于一个元素,而类选择器可以应用于多个元素。ID 选择器以半角"#"开头,格式如下:

```
#one { color : blue; font-size:18px; }
```

要应用 ID 选择器定义的样式,首先必须给某个元素添加 ID 属性。示例代码如下:

```
<style type="text/css">
```

```
#one{
    font-weight:bold;          /* 粗体 */    }
#two{
    font-size:30px;            /* 字体大小 */
    color:#009900;             /* 颜色 */    }
</style>
    <p id="one">ID选择器 1</p>          <!--第 1行 -->
    <p id="two">ID选择器 2</p>
    <p id="two">ID选择器 3</p>
    <p id="one two">ID选择器 3</p>
```

上例中,第 1 行应用了#one 的样式。而第 2 行和第 3 行将同一个 ID 选择器应用到两个元素上,显然违反了一个 ID 选择器只能应用在一个元素上的规定,但浏览器却也显示了 CSS 样式风格且没有报错。尽管如此,我们在编写 CSS 代码时,还是应该养成良好的编码习惯,一个 ID 最多只能赋予一个 HTML 元素,因为每个元素定义的 ID 不只是 CSS 可以调用,JavaScript 等脚本语言也可以调用,如果一个 HTML 文档中有两个相同 ID 属性的元素,那么将导致 JavaScript 在查找 ID 时出错(如 getElementById ()函数)。

第 4 行在浏览器中没有任何 CSS 样式风格显示,这意味着 ID 选择器不支持像 class 选择器那样的多个 ID 名同时使用。因为每个元素和它的 ID 是一一对应的关系,不能为一个元素指定多个 ID,也不能将多个元素定义为一个 ID。类似 id="one two"这样的写法是错误的。

关于类名和 ID 名是否区分大小写,CSS 大体上是不区分大小写的语言,但对于类名和 ID 名是否区分大小写取决于标记语言是否区分大小写,如果使用 XHTML,那么类名和 ID 名是区分大小写的;如果是 HTML,则不区分大小写。另外,ID 名或类名的第一个字符不能为数字。

4.1.4 CSS 文本修饰

文本的美化是网页美观的一个基本要求。通过 CSS 强大的文本修饰功能,可以对文本样式进行更加精细的控制,其功能远比 HTML 中的<font>标记强大。

CSS 中控制文本样式的属性主要有 font-属性类和 text-属性类,再加上修改文本颜色的 color 属性和行高 line-height 属性。Dreamweaver 中这些属性的设置是放在 CSS 规则定义面板的“类型”和“区块”中的。下面是利用 CSS 文本属性对文章进行排版的例子(4-1.html),其显示效果如图 4-2 所示。

```
<style type="text/css">
    h1 {
        font-size: 16px;
        text-align: center;
        letter-spacing: 0.3em;}
    p {
        font-size: 12px;
```

```
            line-height: 160% ;
            text-indent: 2em;}
        .source {
        color: #999999;
        text-align: right;}
</style>
        <h1>失败的权利</h1>
<p class="source">2006 年 5 月 11 日    美国《侨报》</p>
<p>自从儿子进了足球队,……,不亲身经历是无法体会的。</p>
<p>他们队有个传统,……几乎是战无不胜的。</p>
<p>在我看来,……孩子们是当之无愧的。</p>
<p>接受孩子的失败,就给了他成功的机会。</p>
```

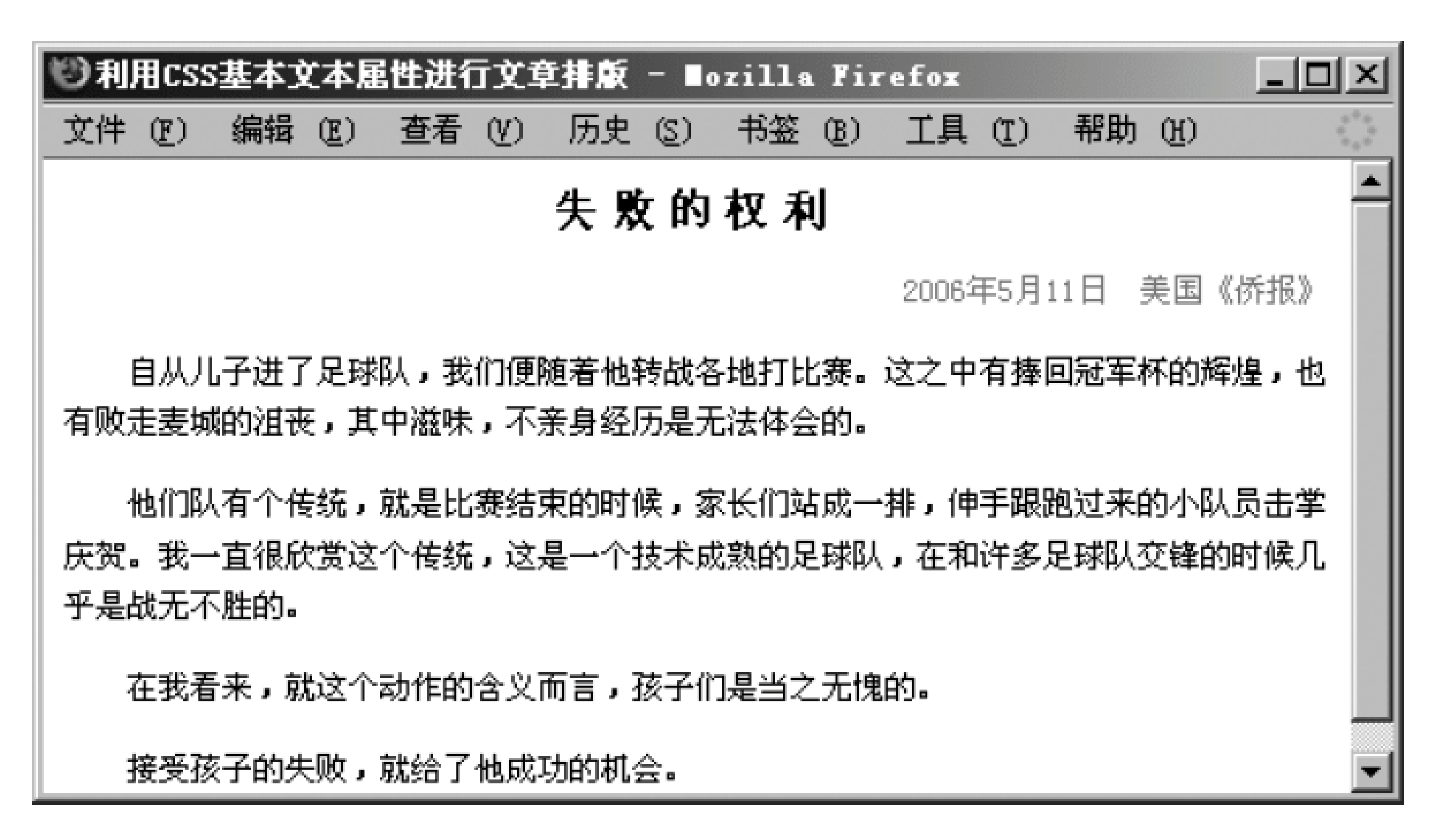


图 4-2　用 CSS 文本属性修饰文本

其中 text-indent 表示首行缩进,在每段开头空两格通常是用 text-indent: 2em 来实现,text-decoration:none 表示去掉下划线,line-height: 160%表示行距为字体高度的 1.6 倍。letter-spacing 用于设置字符间的水平间距。text-align 用于设置文本的对齐方式。

由于大部分 HTML 元素的浏览器默认字体大小是 16px,显得过大;行距是单倍行距,显得过窄。因此制作网页文本时很有必要使用 CSS 文本属性对其进行调整,网页中流行的字体大小有 12px 和 14px,这两种字体大小都比较美观。

如果要设置的字体属性过多,可以使用字体属性的缩写 font,例如"font: 12px/1.6 Arial;"表示 12 像素字体大小,1.6 倍行距,但必须同时定义字体和字号才有效,因此这条规则中定义的字体 Arial 是不能省略的。

4.1.5　伪类选择器及其应用

伪类(pseudo-class)是用来表示动态事件、状态改变或者是在文档中以其他方法不能轻易实现的情况——例如用户的鼠标悬停或单击某元素。总的来说,伪类可以对目标元素出现某种特殊的状态应用样式。这种状态可以是鼠标停留在某个元素上,或者是访问一个超链接。伪类允许设计者自由指定元素在一种状态下的外观。

1. 常见的伪类选择器

常用的伪类有 4 个，分别是：link（链接）、：visited（已访问的链接）、：hover（鼠标悬停状态）和：active（激活状态）。其中前面两个称为链接伪类，只能应用于链接 a 元素，后两种称为动态伪类，理论上可以应用于任何元素，但 IE 6 只支持 a 元素的上述伪类。其他的一些伪类如：focus（获得焦点时的状态）因为在 IE 6 中不支持，所以用得较少。伪类选择器必须指定标记名，且标记和伪类之间用“：”隔开，示例代码如下：

```
a:hover { color : red; text-decoration: underline; }
```

该示例中的伪类选择器作用是定义所有 a 元素在鼠标悬停（hover）状态下的样式。

2. 伪类选择器的应用——制作动态超链接

在默认情况下，网页中的超链接为统一的蓝色带下划线，被单击过的超链接则为紫色带下划线，这种传统的超链接样式看上去过于呆板。

但现在大多数网页中的超链接初始时都没有下划线，并且具有动态效果。例如，当光标移动到超链接上时，超链接会变色并添加下划线等，以提示用户这里可以单击，这样不仅美观而且对用户更友好。而在 HTML 中，只能用<a>标记来表示链接元素，并没有设置超链接在不同状态下样式的方法。

动态超链接是通过 CSS 伪类选择器实现的，因为伪类可以描述超链接在不同状态下的样式，所以我们通过定义 a 元素的各种伪类具有不同的样式，就能制作出千变万化的动态超链接效果。具体来说，<a>标记有 4 种伪类，用来描述链接的 4 种状态，如表 4-2 所示。

表 4-2　超链接<a>标记的 4 个伪类

伪　类	作　　用
a:link	超链接的普通样式风格，即正常浏览状态时的样式
a:visited	被单击过的超链接的样式风格
a:hover	鼠标指针悬停在超链接上时的样式风格
a:active	当前激活（在鼠标单击与释放之间发生）的样式风格

通过 CSS 伪类，只要分别定义上述 4 个状态（或其中几个）的样式代码，就能实现动态超链接效果，如图 4-3 所示。代码如下：

```
<style type="text/css">
a { font-size: 14px;    text-decoration: none; }   /* 设置链接的默认状态 */
a:link { color: #666; }
a:visited { color: #000; }
a:hover { color: #900; text-decoration: underline; background:#9CF; }
a:active { color: #FF3399; }
</style>
```

```
<a href="#">首 页</a><a href="#">系部概况</a><a href="#">联系我们</a>
```



图 4-3 动态超链接

上例中分别定义了链接在 4 种不同的状态下具有不同的颜色，在鼠标悬停时还将添加下划线并改变背景颜色。需要注意的是：

（1）链接伪类选择器的书写应遵循 LVHA 的顺序，即 CSS 代码中 4 个选择器出现的顺序应为 a：link→ a：visited→ a：hover→ a：active，若违反这种顺序，某些样式可能不起作用。

（2）各种 a 的伪类选择器将继承<a>标记选择器定义的样式。

（3）a：link 选择器只能选中具有 href 属性的<a>标记，而 a 选择器能选中所有<a>标记，包括用作锚点的<a>标记。

4.2 CSS 的特性

CSS 具有两大特性：层叠性和继承性。利用这两大特性可大大减少 CSS 代码的编写量。

4.2.1 CSS 的层叠性

所谓层叠性，是指多个 CSS 选择器的作用范围发生了叠加，比如页面中某些元素同时被多个选择器选中（就好像同一个案例适用于多个法律条文一样）。层叠性讨论的问题是：当有多个选择器都作用于同一元素时，CSS 该如何处理？

CSS 的处理原则如下：

（1）如果多个选择器定义的规则未发生冲突，则元素将应用所有选择器定义的样式。例如，下面代码的显示效果如图 4-4 所示。

```
<style type="text/css">
p{                                      /* 标记选择器 */
    color:blue;
    font-size:18px;}
.special{                               /* 类选择器 */
    font-weight: bold;      }
#underline{                             /* ID选择器 */
    text-decoration: underline; }       /* 有下划线 */
</style>
<p>标记选择器 1</p>
<p>标记选择器 2</p>
<p class="special">受到标记、类两种选择器作用</p>
<p id="underline" class="special">受到标记、类和 ID三种选择器作用</p>
```

在代码中，所有 p 元素都被标记选择器 p 选中，同时第 3、4 个 p 元素又被类选择器.special 选中，第 4 个 p 元素还被 ID 选择器 underline 选中，由于这些选择器定义的规则没

有发生冲突，所以被多个选择器同时选中的第 3 和第 4 个元素将应用多个选择器定义的样式。

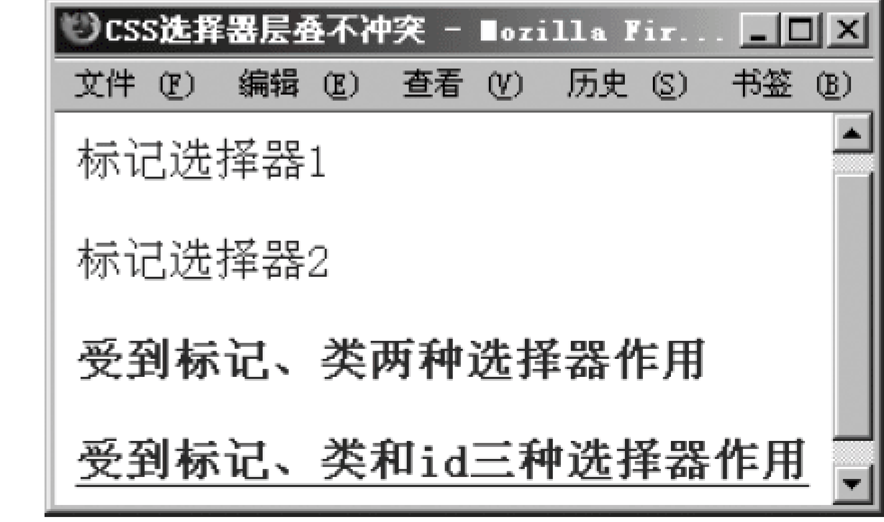


图 4-4 选择器层叠不冲突时的样式

（2）如果多个选择器定义的规则发生了冲突，则 CSS 按选择器的优先级让元素应用优先级高的选择器定义的样式。CSS 规定选择器的优先级从高到低依次为：

行内样式>ID样式>类别样式>标记样式

总的原则是：越特殊的样式，优先级越高。示例代码如下：

```
<style type="text/css">
p{                              /* 标记选择器 */
    color:blue;                 /* 蓝色 */
    font-style: italic;         /* 斜体 */     }
.green{                         /* 类选择器 */
    color:green;                /* 绿色 */     }
.purple{
    color:purple;               /* 紫色 */     }
#red{                           /* ID选择器 */
    color:red;                  /* 红色 */     }
</style>
    <p>这是第 1行文本</p>   <!--蓝色,所有行都以斜体显示 -->
    <p class="green">这是第 2行文本</p>   <!--绿色 -->
    <p class=" green" id="red">这是第 3行文本</p>   <!--红色 -->
    <p id="red" style="color:orange;">这是第 4行文本</p>   <!--黄色 -->
    <p class="purple green">这是第 5行文本</p>   <!--紫色 -->
```

由于类选择器的优先级比标记选择器的优先级高，而类选择器中定义的文字颜色规则和标记选择器中定义的发生了冲突，因此被两个选择器都选中的第 2 行 p 元素将应用 .green 类选择器定义的样式，而忽略 p 选择器定义的规则，但 p 选择器定义的其他规则还是有效的。因此第 2 行 p 元素显示为绿色斜体的文字；同理，第 3 行 p 元素将按优先级高低应用 ID 选择器的样式，显示为红色斜体；第 4 行 p 元素将应用行内样式，显示为黄色斜体；第 5 行 p 元素同时应用了两个类选择器 class="purple green"，两个选择器的优先级相同，则以 class 属性中第一个类名对应的选择器(.purple)为准，显示为紫色斜体。

（3）!important 关键字。

!important 关键字用来强制提升某条声明的重要性。如果在不同选择器中定义的声明发生冲突，而且某条声明后带有!important，则优先级规则为“!important>行内样式>ID 样式>类别样式>标记样式”。对于上例，如果给 .green 选择器中的声明后添加!important，则第 3 行和第 5 行文本都会变为绿色。在任何浏览器中预览都是这种效果。

```
.green{                         /* 类选择器 */
```

```
color:green!important;          /* 通过!important提升该样式的优先级 */
}
```

如果在同一个选择器中定义了两条相冲突的规则，那么 IE 6 总是以最后一条为准，不认!important，而 Firefox/IE 7+以定义了!important 的为准。

```
#box {
    color:red!important;          /* Firefox/IE 7以这一条为准 */
    color:blue;                   /* IE 6总是以最后一条为准 */
    }
```

!important 用法总结：

(1) 在同一选择器中定义的多条样式发生了冲突，则 IE 6 会忽略样式后的!important 关键字，总是以最后定义的那一条样式为准；

(2) 在不同选择器中定义的样式发生冲突，那么所有浏览器都以!important 样式的优先级为最高。

4.2.2　CSS 的继承性

CSS 的继承性是指如果子元素定义的样式没有和父元素定义的样式发生冲突，那么子元素将继承父元素的样式风格，并可以在父元素样式的基础上再加以修改，添加新的样式，而子元素的样式风格不会影响父元素。例如，下面代码的显示效果如图 4-5 所示。

```
<style type="text/css">
body {
    text-align: center;
    font-size: 14px;
    text-decoration: underline;     }
.right{
    text-align: right;   }
p {
    text-decoration:overline;     /* 加上划线 */}
</style>
    <h2>十二星座传说</h2>
    <p><em>白羊座</em>的传说</p>
    <p>天蝎座的传说</p>
    <p class="right">双鱼座的起源</p>
```

说明：

(1) 本例中 body 标记选择器定义的文本居中，14px 字体、带下划线等属性都被所有子元素(h2 和 p)所继承，因此前三行完全应用了 body 定义的样式，而且 p 元素还把它继承的样式传递给了子元素 em，但第四行的 p 元素由于通过“.right”类选择器重新定义了右对齐的样式，所以将覆盖父元素 body 的居中对齐，显示为右对齐。

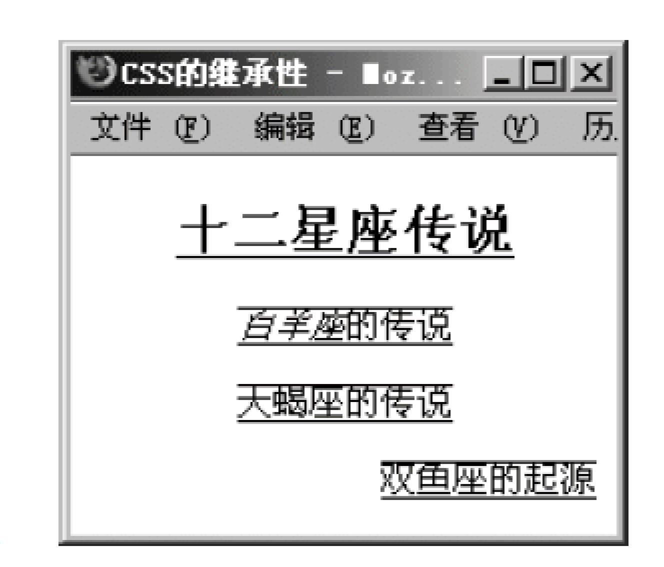


图 4-5　CSS 的继承性示例

（2）由于浏览器对 h2 标题元素预定义了默认样式，该样式覆盖了 h2 元素继承的 <body> 标记选择器定义的 14px 字体样式，结果显示为 h2 元素的字体大小，粗体。可见，继承的样式比元素的浏览器默认样式的优先级还要低。如果要使 h2 元素显示为 14px 大小，需要对该元素直接定义字体大小以覆盖浏览器默认样式。

CSS 的继承贯穿整个 CSS 设计的始终，每个标记都遵循着 CSS 继承的概念。可以利用这种巧妙的继承关系，大大缩减代码的编写量，并提高可读性，尤其在页面内容很多且关系复杂的情况下。例如，如果网页中大部分文字的字体大小都是 12px，我们可以对 body 或 td（若网页用表格布局）标记定义字体样式为 12px。这样由于其他标记都是 body 的子标记，会继承这一样式，就不需要对这么多的子标记分别定义样式了，有些特殊的地方如果字体大小要求是 14px，则可以再利用类选择器或 ID 选择器对它们单独定义。

一个 HTML 文档中元素的继承关系可用如图 4-6 所示的文档对象模型来描述。

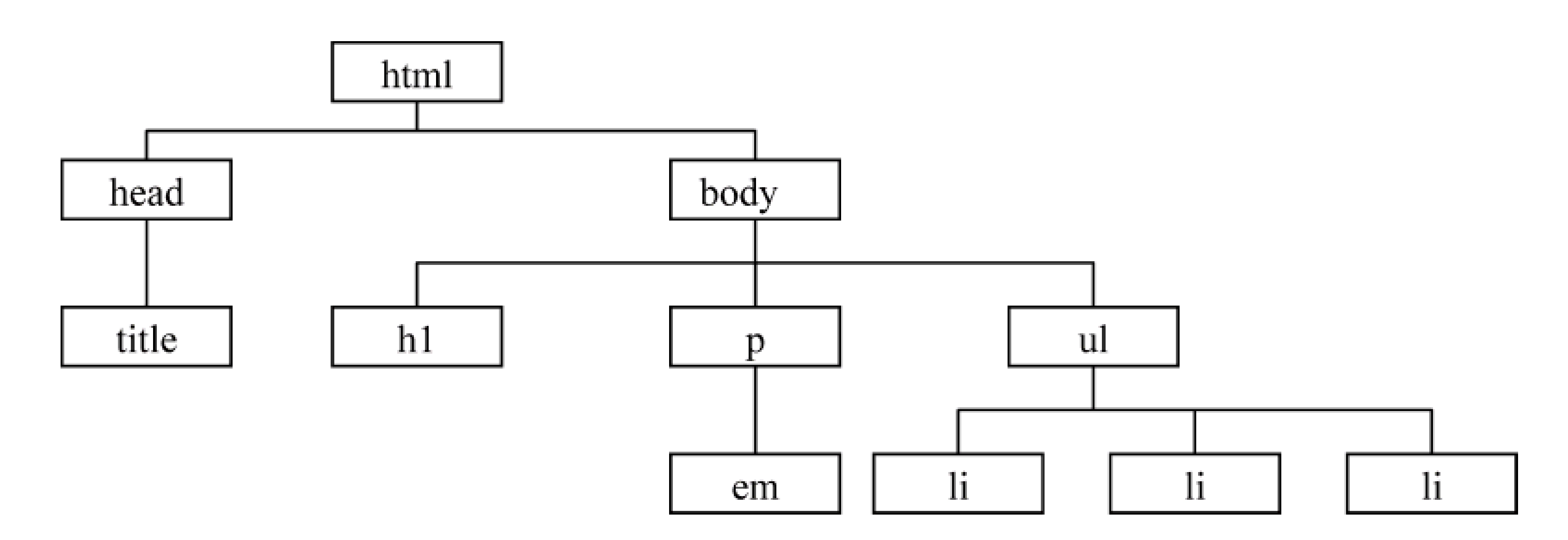


图 4-6　文档对象模型（DOM）图

注意：*并不是所有的 CSS 属性都具有继承性，一般是 CSS 的文本属性具有继承性，而其他属性（如背景属性、布局属性等）则不具有继承性。*

具有继承性的属性大致有：color、font-（以 font 开头的属性）、text-indent、text-align、text-decoration、line-height、letter-spacing、border-collapse 等。

无继承性的属性有：text-decoration：none、所有背景属性、所有盒子属性（边框、边界、填充）、布局属性（如 float）等。要注意的是，text-decoration 属性设置为 none 时不具有继承性，而设置为其他值时又具有继承性。

4.2.3　选择器的组合

每个选择器都有它的作用范围，前面介绍的各种基本选择器，其作用范围都是一个单独的集合，如标记选择器的作用范围是具有该标记的所有元素的集合，类选择器的作用范围是自定义的一类元素的集合。有时我们希望对几种选择器的作用范围取交集、并集、子集以选中需要的元素，这时就要用到复合选择器了，它是通过对几种基本选择器的组合，实现更强、更方便的选择功能。

复合选择器就是两个或多个基本选择器，通过不同方式组合形成的选择器，主要有交集选择器、并集选择器和后代选择器。

1. 交集选择器

交集选择器是由两个选择器直接连接构成，其结果是选中两者各自作用范围的交集。

其中第一个必须是标记选择器，第二个必须是类选择器或 ID 选择器。例如，h1. clas1；p＃intro，这两个选择器之间不能有空格。格式如下：

```
h1.clas1 {color: green; font-size:24px;}
```

交集选择器将选中同时满足前后二者定义的元素，也就是前者定义的标记类型，并且指定了后者类名或 ID 的元素。下面的代码演示了如何应用交集选择器。

```
<style type="text/css">
p {
    color: blue;    }
.special {
    color: green;   }
p.special {
    color: red;     }
</style>
<p>普通段落文本</p>              <!--蓝色-->
<h3>普通 h3标题文本</h3>
<p class="special">指定了 special 类别的段落文本</p>      <!--红色-->
<h3 class="special">指定了 special 类别的 h3标题</h3>    <!--绿色-->
```

上例中<p>标记选择器选中了第一、三行文本；. special 类选择器选中了第三、四行文本，p. special 选择器选中了第三行文本，是两者的交集，用于对段落文本中的第三行进行特殊的控制。因此第三行文本显示为红色，第一行显示为蓝色，第四行显示为绿色。第二行不受这些选择器的影响，仅作对比。

2. 并集选择器

所谓并集选择器，其实就是对多个选择器进行集体声明，多个选择器之间用“，”隔开，其中每个选择器都可以是任意类型的选择器。如果某些选择器定义的样式完全相同，或者部分相同，就可以用并集选择器同时声明这些选择器完全相同或部分相同的样式。

下面的代码演示了并集选择器的作用。

```
<style type="text/css">
    h1,h2,h3,p {
        font-size: 12px;
        background-color:#fcd;    }
    h2.special,.special,#one {
        text-decoration: underline;    }
</style>
    <h1>示例文字 h1</h1>
    <h2 class="special">示例文字 h2</h2>
    <h3>示例文字 h3</h3>
    <h4 id="one">示例文字 h4</h4>
    <p class="special">示例段落 p</p>
```

代码中首先通过集体声明 h1、h2、h3、p 的样式，使 h1、h2、h3、p 选中的第一、二、三、五行的元素都变为紫色，12px 大小，然后再对需要特殊设置的第二、四、五行添加下划线。效果如图 4-7 所示。

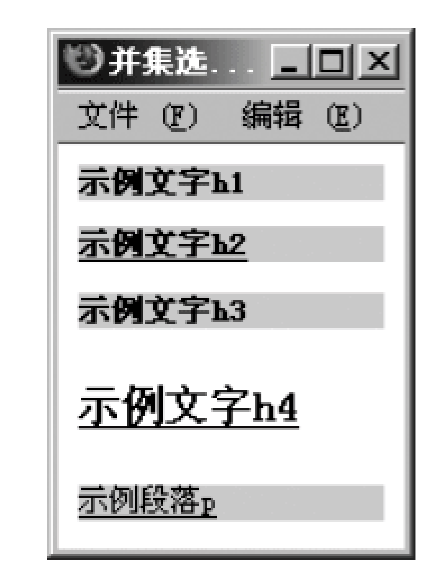


图 4-7 并集选择器示例

3. 后代选择器

在 CSS 选择器中，还可以通过嵌套的方式，对内层的元素进行控制。例如，当 b 元素被包含在 a 元素中时，就可以使用后代选择器 a b{…}选中出现在 a 元素中的 b 元素。后代选择器的写法是把外层的标记写在前面，内层的标记写在后面，之间用空格隔开，下面的代码演示了后代选择器的作用。

```
<style type="text/css">
a {
    font-size: 16px;
    color: red; }
a b {
    color: mediumpurple; }
</style
<b>这是 b标记中的文字</b><br />
<a href="#">这是<b>a标记中的 b<span>标记</span></b></a>
```

其中 a 元素被标记选择器 a 选中，显示为 16px 红色字体；而 a 元素中的 b 元素被后代选择器 a b 选中，颜色被重新定义为淡紫色；第一行的 b 元素未被任何选择器选中。效果如图 4-8 所示。由此可见，后代选择器 a b{…}的选择范围如图 4-9 所示。

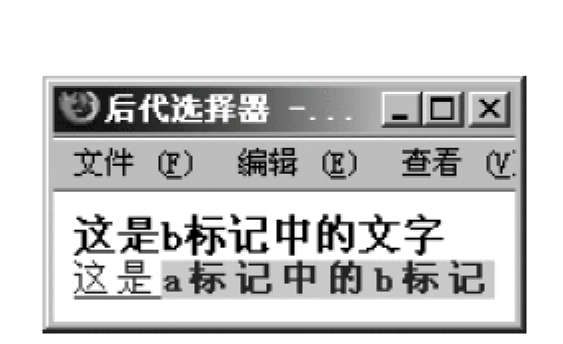


图 4-8 后代选择器示例

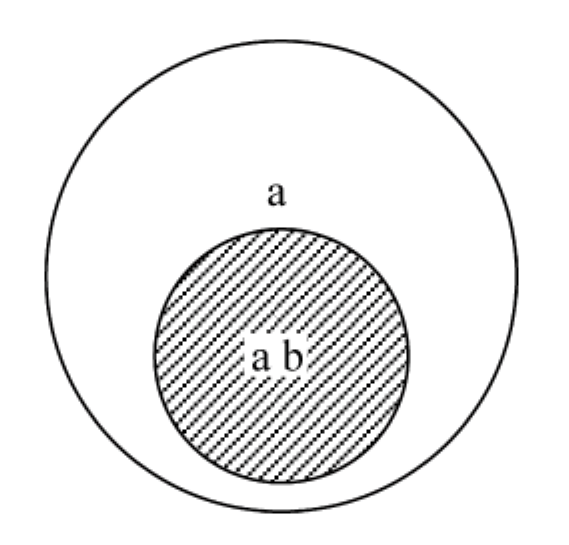


图 4-9 后代选择器的选择范围

同其他 CSS 选择器一样，后代选择器定义的样式同样也能被其子元素继承。例如，在上例中，b 元素内又包含了 span 元素，那么 span 元素也将显示为淡紫色。这说明子元素(span)继承了父元素(a b)的颜色样式。

后代选择器的使用非常广泛，实际上不仅标记选择器可以用这种方式组合，类选择器和 ID 选择器也都可以进行嵌套，而且后代选择器还能够进行多层嵌套。例如：

```
.special b { color: red }                    /* 应用了类 special 的元素里面包含的<b> */
#menu li { padding: 0 6px; }                 /* ID为 menu 的元素里面包含的<li> */
td.top .banl strong{ font-size: 16px; }      /* 多层嵌套,同样适用 */
#menu a:hover b          /* ID为 menu 的元素里的 a:hover 伪类里包含的<b> */
```

提示：选择器的嵌套在 CSS 的编写中可以大大减少对 class 或 id 的定义。因为在构建 HTML 框架时通常只需给父元素定义 class 或 id，子元素能通过后代选择器选择的，则利用这种方式，而不需要再定义新的 class 或 id。

4. 复合选择器的优先级

复合选择器的优先级比组成它的单个选择器的优先级都要高。我们知道，基本选择器的优先级是“ID 选择器＞类选择器＞标记选择器”，所以不妨设 ID 选择器的优先级权重是 100，类选择器的优先级权重是 10，标记选择器的优先级权重是 1，那么复合选择器的优先级就是组成它的各个选择器权重值的和。例如：

```
h1{color:red;}                    /* 权重=1 */
p em{color:blue;}                 /* 权重=2 */
.warning{color:yellow;}           /* 权重=10 */
p.note em.dark{color:gray;}       /* 权重=22 */
#main{color:black;}               /* 权重=100 */
```

当权重值一样时，会采用“层叠原则”，一般后定义的会被应用。

下面是复合选择器优先级计算的一个例子。

```
<style type="text/css">
    #aa ul li {  color:red  }
    .aa {  color:blue  }
</style>
<div id="aa">
    <ul><li class="aa">
        CSS 常见问题之<em class="aa">复合选择器</em>的优先级
        </li>
    </ul></div>
```

对于<li>标记中的内容，它同时被“#aa ul li”和“.aa”两个选择器选中，由于#aa ul li 的优先级为 102，而.aa 的优先级为 10，所以 li 中的内容将应用#aa ul li 定义的规则，文字为红色，如果希望文字为蓝色，可提高.aa 的特殊性，将其改写成“#aa ul li.aa”。

另外，代码中 em 元素内的文字颜色为蓝色，因为直接作用于 em 元素的选择器只有“.aa”，虽然 em 也会继承“#aa ul li”选择器的样式，但是继承的样式优先级最低，会被类选择器“.aa”定义的样式所覆盖。

综上所述，CSS 样式的优先级如图 4-10 所示。

其中，浏览器对标记预定义的样式是指对于某些 HTML 标记，浏览器预先对其定义了默认的 CSS 样式，如果用户没重新定义样式，那么浏览器将按其定义的默认样式显示，常见的标记在标准浏览器(如 Firefox)中默认样式如下：

```
body { margin: 8px; line-height: 1.12em }
h1 { font-size: 2em; margin: .67em 0 }
h2 { font-size: 1.5em; margin: .75em 0 }
```

```
h3 { font-size: 1.17em; margin: .83em 0 }
h4, p,blockquote, ul,fieldset, form,ol, dl, dir,menu { margin: 1.12em 0 }
h1, h2, h3, h4,h5, h6, b,strong { font-weight: bolder }
blockquote { margin-left: 40px; margin-right: 40px }
pre   { white-space: pre }
```

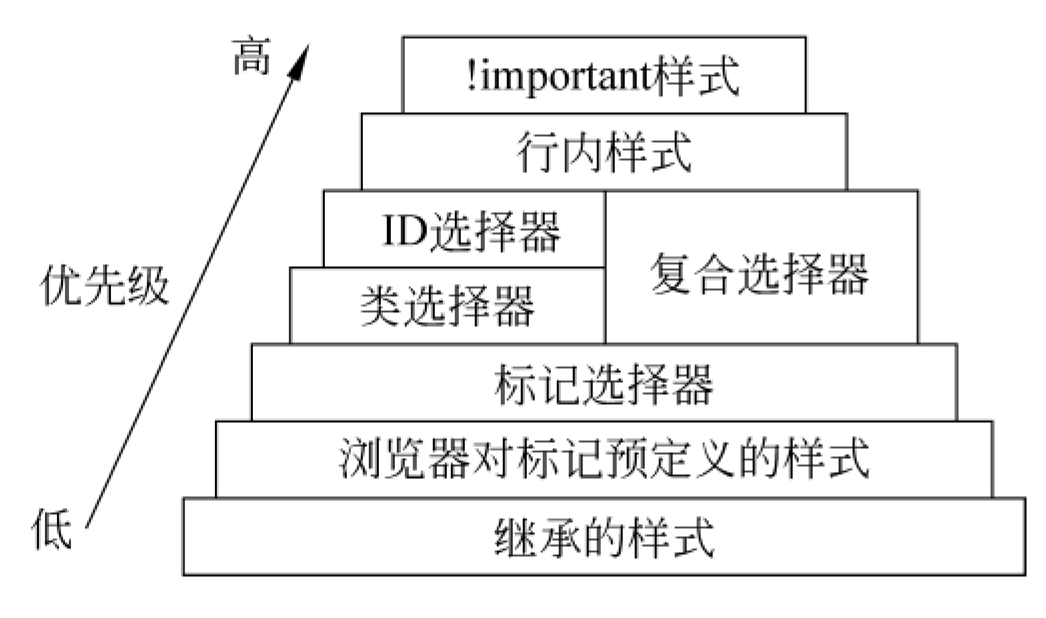


图 4-10　CSS 样式的优先级

有些元素的预定义(默认)的样式在不同的浏览器中区别很大,例如,ul、ol 和 dd 等列表元素,IE 中的默认样式是:

```
ul,ol,dd{margin-left:40px;}
```

而 Firefox 中的默认样式定义为:

```
ul,ol,dd {padding-left:40px;}
```

因此,要清除列表的默认样式,一般可以设置:

```
ul, ol, dd {
    list-style-type:none;          /* 清除列表项目符号 */
    margin:0;                      /* 清除 IE左缩进 */
    padding:0;                     /* 清除非 IE左缩进 */
        }
```

4.2.4　CSS 2.1 新增选择器简介

上面介绍的一些基本选择器和复合选择器都是 CSS 1.0 中就已具有的选择器,它们能被目前所有的浏览器所支持。CSS 2.1 标准在 1.0 的基础上增加了一些新的选择器,这些选择器不能被 IE 6 浏览器支持,但是其他浏览器(如 IE 7+、Firefox、Safari 等)均对它们提供支持,考虑到大多数计算机都安装了 IE 7+等新型浏览器,预计 IE 6 将在一两年内被淘汰,因此我们有必要知道这些新选择器,它们能给 CSS 设计带来方便,而且对以后学习 jQuery 的选择器也是很有帮助的。

1. 子选择器

子选择器用于选中元素的直接后代(即儿子),它的定义符号是大于号(>),例如:

```
body>p {
```

```
    color: green;      }
<body>
    <p>这一段文字是绿色</p>
    <div><p>这一段文字不是绿色</p></div>
    <p>这一段文字是绿色</p>
</body>
```

只有第一个和第三个段落的文字会变绿色，因为它们是 body 元素的直接后代，所以被选中。而第二个 p 元素是 body 的间接后代，不会被选中，如果把（body＞p）改为后代选择器（body p），那么三个段落都会被选中。这就是子选择器和后代选择器的区别。后代选择器选中任何后代。

2. 相邻选择器

相邻（adjacent-sibling）选择器是另一个有趣的选择器，它的定义符号是加号（＋），相邻选择器将选中紧跟在它后面的一个兄弟元素（这两个元素具有共同的父元素）。例如：

```
h2+p {
    color: red;      }
<h2>下面哪些文字是红色的呢</h2>
<p>这一段文字是红色</p>
<p>这一段文字不是红色</p>
<h2>下面有文字是红色的吗</h2>
<div><p>这一段文字不是红色</p></div>          <!--div 中的 p 和 h2 不同级,不会被选中-->
<p>这一段文字不是红色</p>       <!--没有紧跟在 h2 后,不会被选中-->
<h2>下面哪些文字是红色的呢</h2>
这一段文字不是红色
<p>这一段文字是红色</p>
<p>这一段文字不是的</p>
```

第一个段落标记紧跟在 h2 之后，因此会被选中，在最后一个 h2 元素后，尽管紧接的是一段文字。但那些文字不属于任何标记，因此紧随这些文字之后的第一个 p 元素也会被选中。

如果希望紧跟在 h2 后面的任何元素都变成红色，可使用如下方法：

```
h2+ *  {
    color: red;      }
```

那么第二个 h2 后的 div 元素也会被选中。

3. 属性选择器

引入属性选择器后，CSS 变得更加复杂、准确、功能强大。属性选择器主要有三种形式，分别是匹配属性、匹配属性和值、匹配属性和值的一部分。属性选择器的定义方式是将属性和值写在方括号（[ ]）内。

1）匹配属性

属性选择器选中具有某个指定属性的元素，例如：

```
a[name] {color:purple; }                    /* 选中具有 name 属性的 a 元素 */
img[border] {border-color:gray;}            /* 选中具有 border 属性的 img 元素 */
[special] {color:red;}                      /* 选中具有 special 属性的任何元素 */
```

在这些情况下，每个元素的具体属性值并不重要，只要给定属性在元素中出现，元素便匹配该属性选择器，还可给元素自定义一个它没有的属性名，如(<h2 special="">…</h2>)，那么这个 h2 元素会被[special]属性选择器选中，这时属性选择器就起到类或 ID 选择器的作用了。

2）匹配属性和值

属性选择器也可根据元素具有的属性和值来匹配，例如：

```
a[href="http://www.hynu.cn"] {color:yellow;}                  /* 选中指向 www.hynu.cn 的链接 */
input[type="submit"] {background:purple;}            /* 选中表单中的提交按钮 */
img[alt="Sony Logo"][class="pic"] {margin:20px;}     /* 同时匹配两个属性和值 */
```

这样，用属性选择器就能很容易地选中某个特定的元素，而不用为这个特定的元素定义一个 ID 或类，再用 ID 或类选择器去匹配它了。

3）匹配单个属性值

如果一个属性的属性值有多个，每个属性值用空格分开，那么就可以用匹配单个属性值的属性选择器来选中它们了。它是在等号前加了一个波浪符(~)。例如：

```
[special~="wo"] {color: red;}
<h2 special="wo shi">文字是红色</h2>
```

由于对一个元素可指定多个类名，匹配单个属性值的选择器就可以选中具有某个类名的元素，这才是它的主要用途。例如：

```
h2[class~="two"] {color: red;}
<h2 class="one two three">文字是红色</h2>
```

4. 新增加的伪类选择器

在 IE 6 中，只支持<a>标记的四个伪类，即 a:link、a:visited、a:hover 和 a:active，其中前两个称为链接伪类，后两个是动态伪类。在 CSS 2.1 规范中，任何元素都支持动态伪类，所以像 li:hover、img:hover、div:hover 和 p:hover 这些伪类是合法的，它们都能被 IE 7 和其他浏览器支持。

下面介绍两种新增加的伪类选择器，即:focus 和:first-child。

1）:focus

:focus 用于定义元素获得焦点时的样式。例如，对于一个表单来说，当光标移动到某个文本框内时(通常是单击了该文本框或使用 Tab 键切换到了这个文本框上)，这个 input 元素就获得了焦点。因此，可以通过 input:focus 伪类选中它，改变它的背景色，使

它突出显示,代码如下:

```
input:focus { background: yellow; }
```

对于不支持:focus 伪类的 IE 6 浏览器,要模拟这种效果,只能使用两个事件结合 JavaScript 代码来模拟,它们是 onfocus(获得焦点)和 onblur(失去焦点)事件。

2) :first-child

:first-child 伪类选择器用于匹配它的父元素的第一个子元素,也就是说这个元素是它父元素的第一个儿子,而不管它的父元素是哪个。例如:

```
p:first-child{ font-weight: bold; }
<body>
<p>这一段文字是粗体</p>                    <!--第 1行,被选中-->
<h2>下面哪些文字是粗体的呢</h2>
<p>这一段文字不是粗体</p>
<h2>下面哪些文字是粗体的呢</h2>
<div><p>这一段文字是粗体</p>               <!--第 5行,被选中-->
<p>这一段文字不是粗体</p></div>
<div>下面哪些文字是粗体的呢
这一段文字不是
<p>这一段文字是粗体</p>                    <!--第 9行,被选中-->
<p>这一段文字不是的</p></div>
</body>
```

这段文字共有 3 行会以粗体显示。第 1 行 p 是其父元素 body 的第一个儿子,被选中;第 5 行 p 是父元素 div 的第一个儿子,被选中,第 9 行 p 也是父元素 div 的第一个儿子,也被选中,尽管它前面还有一些文字,但那不是元素。

5. 伪对象选择器

在 CSS 中伪对象选择器主要有:first-letter、:first-line 以及:before 和:after。之所以称:first-letter 和:first-line 是伪对象,是因为它们在效果上使文档中产生了一个临时的元素,这是应用"虚构标记"的一个典型实例。

1) :first-letter

:first-letter 用于选中元素内容中的首个字符,:first-line 用于选中元素内容中的首行文本。不管元素显示的区域是宽还是窄,样式都会准确地应用于首行。如果段落的首行只有 5 个汉字,则只有这 5 个汉字会应用样式。如果首行包含 30 个汉字,那么这 30 个汉字都会应用样式。下面是一个 p 元素的 CSS 代码,其显示效果如图 4-11 所示。

```
p:first-letter { font-size: 2em; float: left; }
p:first-line { font-weight: bold; letter-spacing: 0.3em; }
<p>春天来临,又到了播种耕种的季节,新皇后将炒熟了的麦子……</p>
```

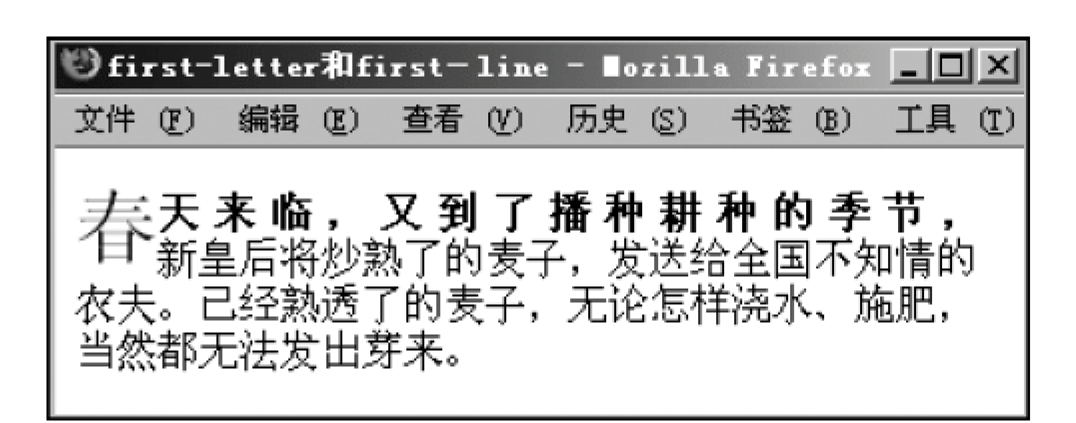


图 4-11　:first-letter 和:first-line 的应用

注意：

(1) 可供:first-line 使用的 CSS 属性有一些限制，它只能使用字体、文本和背景属性，不能使用盒子模型属性(如边框、背景)和布局属性。

(2) 如果在:first-letter 或:first-line 与"{"之间有空格，则 IE 6 也能支持这两种选择器。

2) :before 和:after

:before 和:after 两个伪对象必须配合 content 属性使用才有意义。它们的作用是在指定的元素内产生一个新的行内元素，该行内元素的内容是由 content 属性里的内容决定。例如，下面代码的效果如图 4-12 所示。

```
<style>
    p:before, p:after{content: "--"; color:red;}
</style>
<p>看这一段文字的左右</p>
<p>这一段文字左右</p>
```

可以看到，通过产生内容属性，p 元素的左边和右边都添加了一个新的行内元素，它们的内容是"--"，并且设置伪元素内容的样式红色。

还可以将:before 和:after 伪元素转化为块级元素显示，例如将上述选择器修改为：

```
p:before,p:after{content: "--"; color:red; display:block;}
```

显示效果如图 4-13 所示。利用:after 产生的伪元素，常被用来作为清除浮动的元素，这样就不需要在浮动元素后添加一个空元素了。具体请参考 4.6.3 节相关内容。

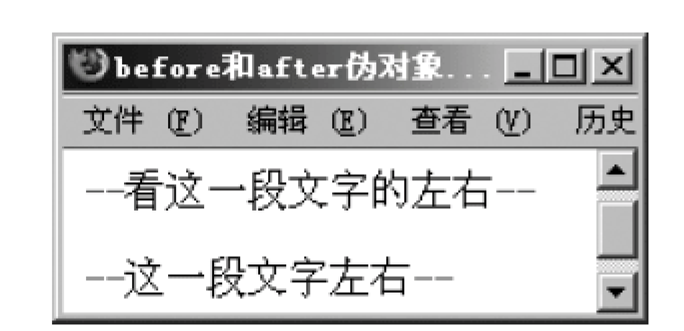


图 4-12　用:before 和:after 配合 content 添加伪元素

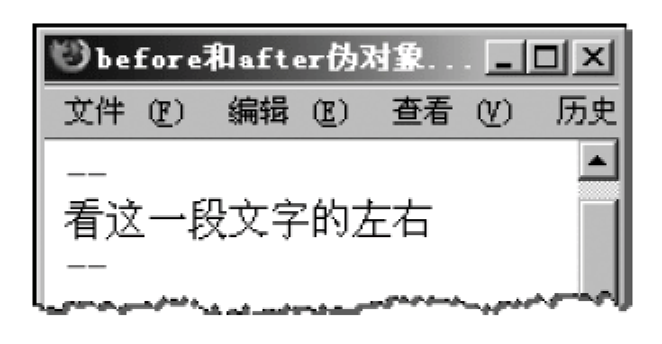


图 4-13　设置伪元素为块级元素显示的效果

6. CSS 2.1 选择器总结

下面将常用的 CSS 2.1 选择器罗列在表 4-3 中，请读者掌握它们的用法。

表 4-3 CSS 2.1 常用的选择器

选择器名称	选择器示例	作 用 范 围
通配选择符	*	所有的元素
标记选择器	div	所有<div>标记的元素
后代选择器	div*	<div>标记中所有的子元素
	div span	包含在<div>标记中的 span 元素
	div .class	包含在<div>标记中类名属性为 class 的元素
并集选择器	div, span	div 元素和 span 元素
子选择器*	div>span	如果 span 元素是 div 元素的直接后代，则选中 span 元素
相邻选择器*	div+span	如果 span 元素紧跟在 div 元素后，则选中 span 元素
类选择器	.class	所有类名属性为 class 的元素
交集选择器	div.class	所有类名属性为 class 的 div 元素
ID 选择器	#itemid	ID 名为 itemid 的唯一元素
	div#itemid	ID 名为 itemid 的唯一 div 元素
属性选择器*	a[attr]	具有 attr 属性的 a 元素
	a[attr='x']	具有 attr 属性并且值为 x 的 a 元素
	a[attr~='x']	具有 attr 属性并且值的字符中含有'x'的 a 元素
伪类选择器	a:hover	所有在 hover 状态下的 a 元素
	a.class:hover	所有在 hover 状态下具有 class 类名的 a 元素
伪对象选择器*	div:first-letter	选中 div 元素中的第一个字符

4.3 CSS 设计和书写技巧*

4.3.1 CSS 样式总体设计原则

设计 CSS 样式时，应遵循“先普遍、后特别”的原则。首先对很多元素统一设置属性，然后为一些需要特别设置样式的元素添加 class 属性或 ID 属性，并注意如下几点：

(1) 善于运用后代选择器。虽然定义标记选择器最方便(不需要在每个标记中添加 class 或 ID 属性，使初学者最喜欢定义标记选择器或由标记选择器组成的后代选择器)，但有些标记在网页文档的各部分出现的含义不同，从而样式风格往往也不相同，例如，网页中普通的文字链接和导航链接的样式就不同。为此，虽然可以将导航条内的各个<a>标记都定义为同一个类，但这样导航条内的所有<a>标记都得添加一个 class 属性，class="nav"要重复写很多遍。例如：

```
<div>
    <a class="nav" href="#">首 页</a>
```

```
        <a class="nav" href="#">中心简介</a>…
        <a class="nav" href="#">技术支持</a></div>
```

实际上，可以为导航条内<a>标记的父标记（如 ul）添加一个 ID 属性（nav），然后用后代选择器（#nav a）就可以选中导航条内的各个<a>标记了。这时 HTML 结构代码中的 ID="nav"就只要写一次了，示例代码如下，显然这样代码更简洁。

```
<div id="nav">
    <a href="#">首 页</a>
    <a href="#">中心简介</a>…
    <a href="#">技术支持</a></div>
```

（2）灵活运用 class 和 ID。

例如，网页中有很多栏目框，所有栏目框有许多样式是相同的，因此可以将所有栏目框都定义为同一个类，然后再对每个栏目框定义一个 ID 属性，以便对某个栏目框作特别的样式设置。

（3）对于几个不同的选择器，如果它们有一些共同的样式声明，就可以先用并集选择器对它们集体声明，然后再单独定义某些选择器的特殊样式以覆盖前面的样式。如：

```
h2,h3,h4,p,form,ul{margin:0;font-size:14px;}
h2{font-size:18px;}
```

4.3.2 Dreamweaver 对 CSS 的可视化编辑支持

1. 新建和编辑 CSS 样式

Dreamweaver 对 CSS 代码的新建和编辑有很好的支持，对 CSS 的所有操作都集中在如图 4-14 中所示的“CSS 样式”面板中，单击“新建 CSS 规则”按钮（），就会弹出如图 4-15 所示的对话框：



图 4-14 CSS 样式面板

其中“选择器类型”中的“类”对应类选择器，“标签”对应标记选择器，“高级”对应除此之外的其他所有选择器（如 ID 选择器，伪类选择器和各种复合选择器）。确定选择器类型后，就可以在“名称”下拉列表框中输入或选择选择器的名称（要注意符合选择器的命名规范，即类选择器必须以点开头，ID 选择器必须以＃开头），在“定义在”选项中，可以选择将 CSS 代码写在外部 CSS 文件中（如 style.css），并通过链接式引入该 CSS 文档；“仅对该文档”表示使用嵌入式引入 CSS，即把 CSS 代码作为<style>标记的内容写在文档头部。

定义好选择器后，单击“确定”按钮，就会弹出该选择器的 CSS 属性面板，如图 4-16 所示。所有选择器的 CSS 属性面板都是相同的。

对面板中任何一项进行赋值后，都等价于往该选择器中添加一条声明，如下划线设置为“无”，就相当于在代码视图内为该选择器添加了一条“text-decoration：none；”。

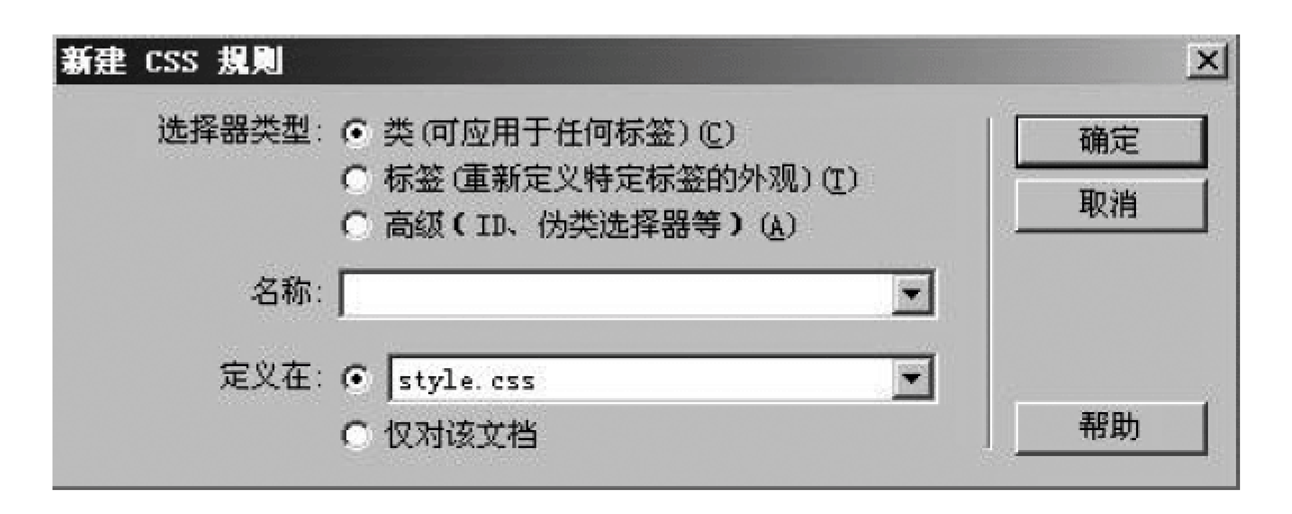


图 4-15　新建 CSS 选择器



图 4-16　CSS 属性面板

设置完样式属性后，单击"应用"按钮，可以在设计视图中看到样式应用后的效果，也可单击"确定"按钮，将关闭规则定义面板并应用样式。这时在"CSS 样式"面板中将出现刚才新建的 CSS 选择器名称和其属性，如图 4-14 所示。

2. 将嵌入式 CSS 转换为外部 CSS 文件

如果在 HTML 文档头部已经用<style>标记添加了一段嵌入式的 CSS 代码，可以将这段代码导出成一个 CSS 文件，供多个 HTML 文档引用。导出方法有以下两种：

(1) 执行"文本"→"CSS 样式"→"导出"菜单命令，输入文件名(如 style.css)，就可将该段 CSS 代码导出成一个.css 文件。导出后可将此文档中的<style>标记部分全部删除，然后再单击图 4-14 中的"附加样式表"按钮( )，将刚才导出的.css 文件引入，引入的方法可选择"链接"或"导入"，分别对应链接式 CSS 或导入式 CSS。

(2) 直接复制 CSS 代码。在 DW 中新建一个 CSS 文件，将<style>标记中的所有样式规则(不包括<style>标记和注释符)剪切到 CSS 文档中，然后再单击"附加样式表"按钮( )将这个 CSS 文件导入。

3. Dreamweaver 对 CSS 样式的代码提示功能

Dreamweaver 对 CSS 同样具有很好的代码提示功能。在代码视图中编写 CSS 代码时，按 Enter 键或空格键都可以触发代码提示。

编辑 CSS 代码时，在一条声明书写结束的地方按 Enter 键，就会弹出该选择器拥有

的所有 CSS 属性列表供选择，如图 4-17 所示。当在属性列表框中已选定某个 CSS 属性后，又会立刻弹出属性值列表框供选择，如图 4-18 所示。如果属性值是颜色，则会弹出颜色选取框；如果属性值是 URL，则会弹出文件选择框。

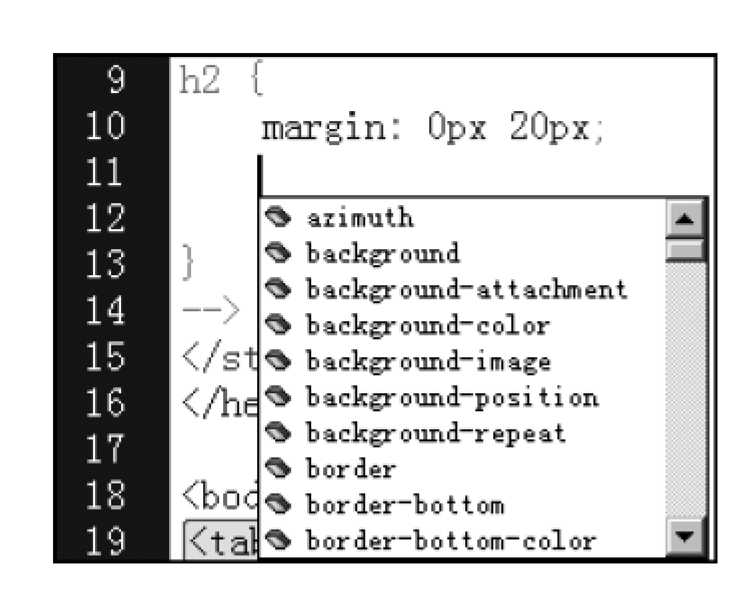


图 4-17　按回车后提示属性名称

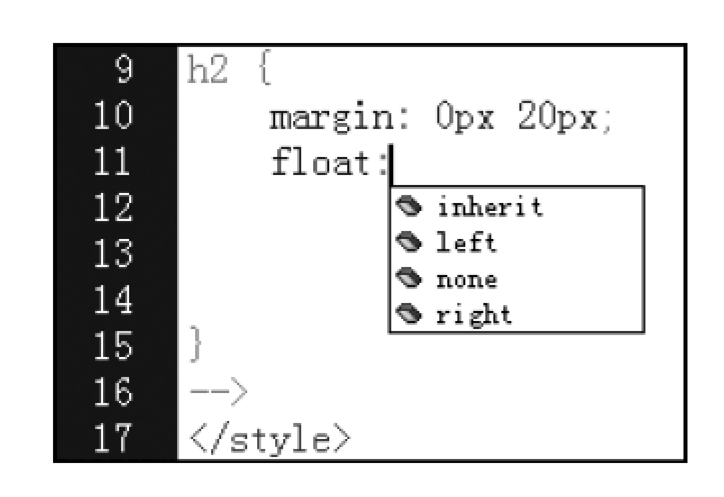


图 4-18　选择名称后提示属性值

如果要修改某个 CSS 属性的值，只需把冒号和属性值删除，然后输入一个冒号，就又会弹出如图 4-18 所示的属性值列表框来。

4. 在代码视图中快速新建选择器和修改选择器

在代码视图中，如果将光标移动到某个标记的标记范围内（尖括号内），如图 4-19 所示。再单击图 4-14 中 CSS 样式面板中的“新建”按钮，则在弹出如图 4-20 所示的“新建 CSS 规则”对话框中，会自动为光标所在位置的元素建立选择器名，这样可免去手工书写该 CSS 选择器的名称的麻烦。

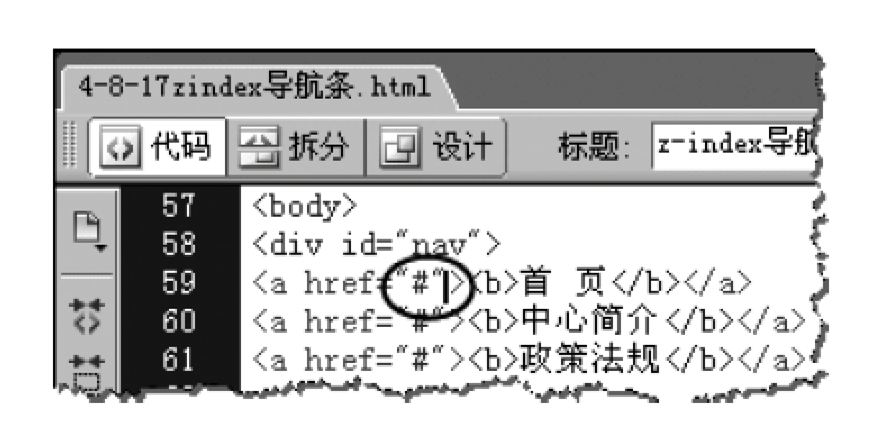


图 4-19　将光标置于标记范围内

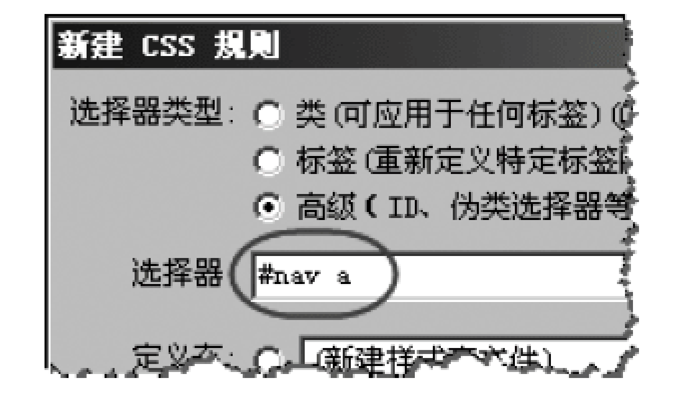


图 4-20　新建选择器时会自动出现光标位置的元素

如果要修改某个 CSS 选择器的样式，可将光标置于这个 CSS 选择器的代码范围内，再单击如图 4-14 所示“CSS 样式”面板中的“编辑样式”按钮，就会弹出该选择器的规则定义面板（见图 4-16）供修改。

4.3.3　CSS 属性的值和单位

值是对属性的具体描述，而单位是值的基础。没有单位，浏览器将不知道一个边框是 10cm 还是 10px。CSS 中较复杂的值和单位有颜色取值和长度单位。

1. 颜色的值

CSS 中定义颜色的值可使用命名颜色、RGB 颜色和十六进制颜色三种方法。

1） 命名颜色

例如：

```
p{ color: red; }
```

其中 red 就是命名颜色,能够被 CSS 识别的颜色名大约有 140 种。常见的颜色名如 red、yellow、blue、silver、teal、white、navy、orchid、oliver、purple、green 等。

2）RGB 颜色

显示器的成像原理是红(Red)、绿(Green)、蓝(Blue)三色光的叠加形成各种各样的色彩。因此,通过设定 RGB 三色的值来描述颜色是最直接的方法。格式如下:

```
td{ color: rgb(139,31,185); }
td{ color: rgb(12% ,201,50% ); }
```

其值可以取 0～255 之间的整数,也可以是 0～100%的百分数,但 Firefox 浏览器并不支持百分数值。

3）十六进制颜色

十六进制颜色的使用最普遍,其原理同样是 RGB 色,不过将 RGB 颜色的数值转换成了十六进制的数字,并用更加简单的方式写出来:#RRGGBB,如#ffcc33。

其参数取值范围为:00～FF(对应十进制仍为 0～255),如果每个参数各自在两位上的数值相同,那么该值也可缩写成"#RGB"的方式。例如,#ffcc33 可以缩写为#fc3。

2. CSS 长度单位

为了正确显示网页中的元素,许多 CSS 属性都依赖于长度。所有长度都可以为正数或者负数加上一个单位来表示,而长度单位大致可分为三类:绝对单位、相对单位和百分比。

1）绝对单位

绝对单位很简单,包括英寸(in)、厘米(cm)、毫米(mm)、磅(pt)和 pica(pc)。

使用绝对单位定义的长度在任何显示器中显示的大小都是相同的,不管该显示器的分辨率或尺寸是多少。如 font-size:9pt,则该文字在任何显示器中都是 9 磅大小。在手机网页中,由于不同类型的手机分辨率相差很大,应尽量使用绝对单位。

2）相对单位

顾名思义,相对单位的长短取决于某个参照物,如屏幕的分辨率、字体高度等。

有 3 种相对长度单位:元素的字体高度(em)、字母 x 的高度(ex)和像素(px)。

(1) em 就是元素原来给定的字体 font-size 的值,如果元素原来给定的 font-size 值是 14px,那么 1em 就是 14px。

(2) ex 是以字体中小写 x 字母为基准的单位,不同的字体有不同的 x 高度,因此即使 font-size 相同而字体不同的话,1ex 的高度也会不同。

(3) 像素 px 是指显示器按分辨率分割得到的小点。显示器由于分辨率或大小不同,像素点的大小是不同的,所以像素也是相对单位。

3）百分比

百分比显得非常简单,也可看成是一个相对量。如:

```
td{font-size:12px; line-height: 160% ; }      /* 设定行高为字体高度的 160% */
```

```
hr{ width: 80% }                    /* 水平线宽度相对其父元素宽度为 80% */
```

4.4 盒子模型及标准流下的定位

在网页的布局和页面元素的表现方面，要掌握的最重要的概念是 CSS 的盒子模型(Box Model)以及盒子在浏览器中的排列(定位)，这些概念用来控制元素在页面上的排列和显示方式，形成 CSS 的基本布局。

设想有 4 幅镶嵌在画框中的画，如图 4-21 所示。我们可以把这 4 幅画看成是 4 个 img 元素，那么 img 元素中的内容就是画框中的画，画(内容)和边框之间的距离称为盒子的填充或内边距(padding)，画的边框称为盒子的边框(border)，画的边框周围还有一层边界(margin)，用来控制元素盒子与其他元素盒子之间的距离。

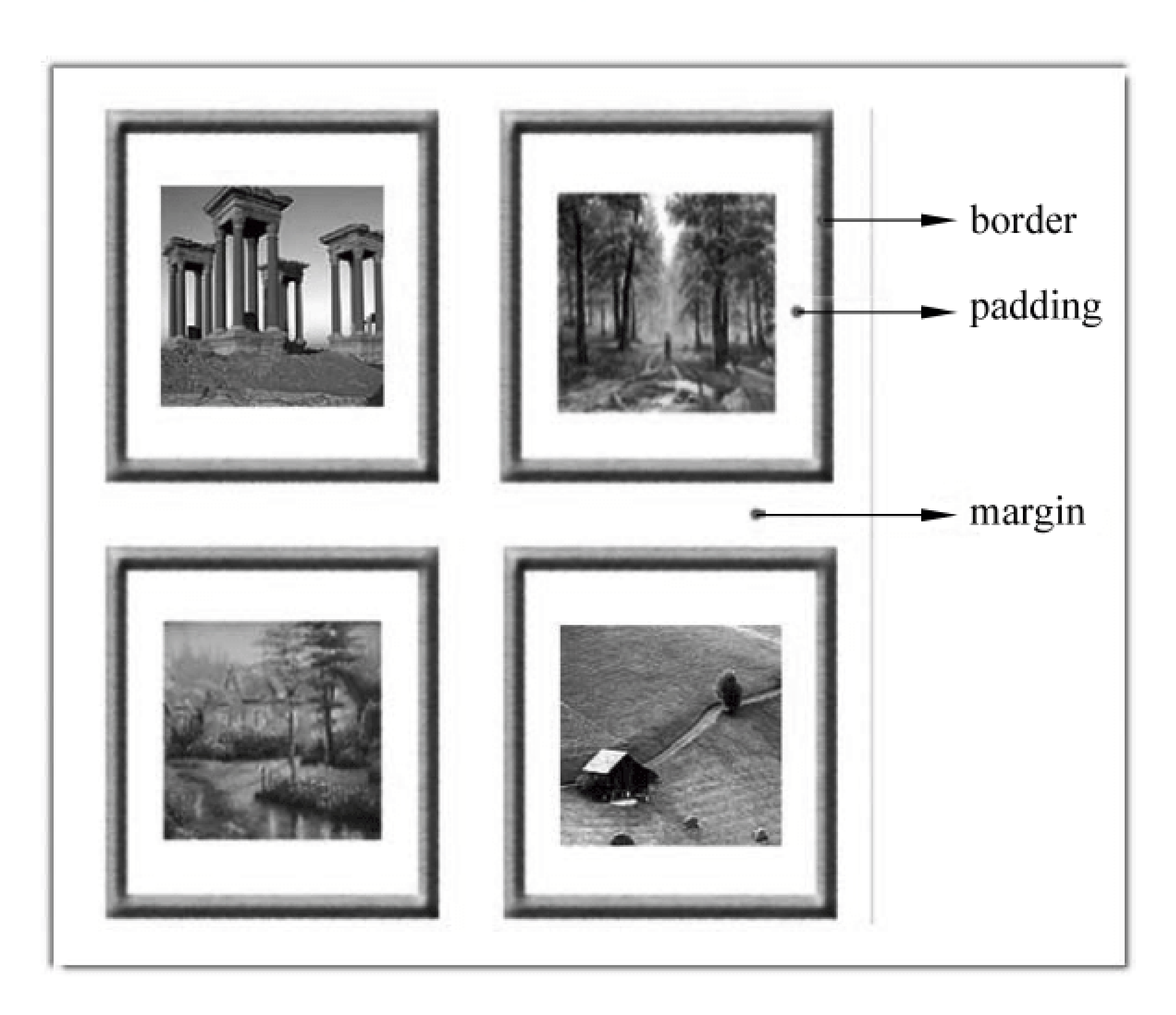


图 4-21 画框示意图

4.4.1 盒子模型基础

通过对画框中的画进行抽象，就得到一个抽象的模型——盒子模型，如图 4-22 所示。盒子模型是 CSS 的基石之一，它指定元素如何显示以及(在某种程度上)如何相互交互，页面上的每个元素都被浏览器看成是一个矩形的盒子，这个盒子由元素的内容、填充、边框和边界组成。网页就是由许多个盒子通过不同的排列方式(上下排列、左右排列、嵌套排列)堆积而成。

1. 盒子模型的属性和计算

盒子的概念是非常容易理解的，但是如果要精确地利用盒子模型布局，有时候 1 像素都不能够差，这就需要非常精确地理解盒子大小的计算方法。盒子模型的填充、边框、边界宽度都可以通过相应的属性分别设置上、右、下、左四个距离的值，内容区域的宽度可通过 width 和 height 属性设置，增加填充、边框和边界不会影响内容区域的尺寸，但会增加

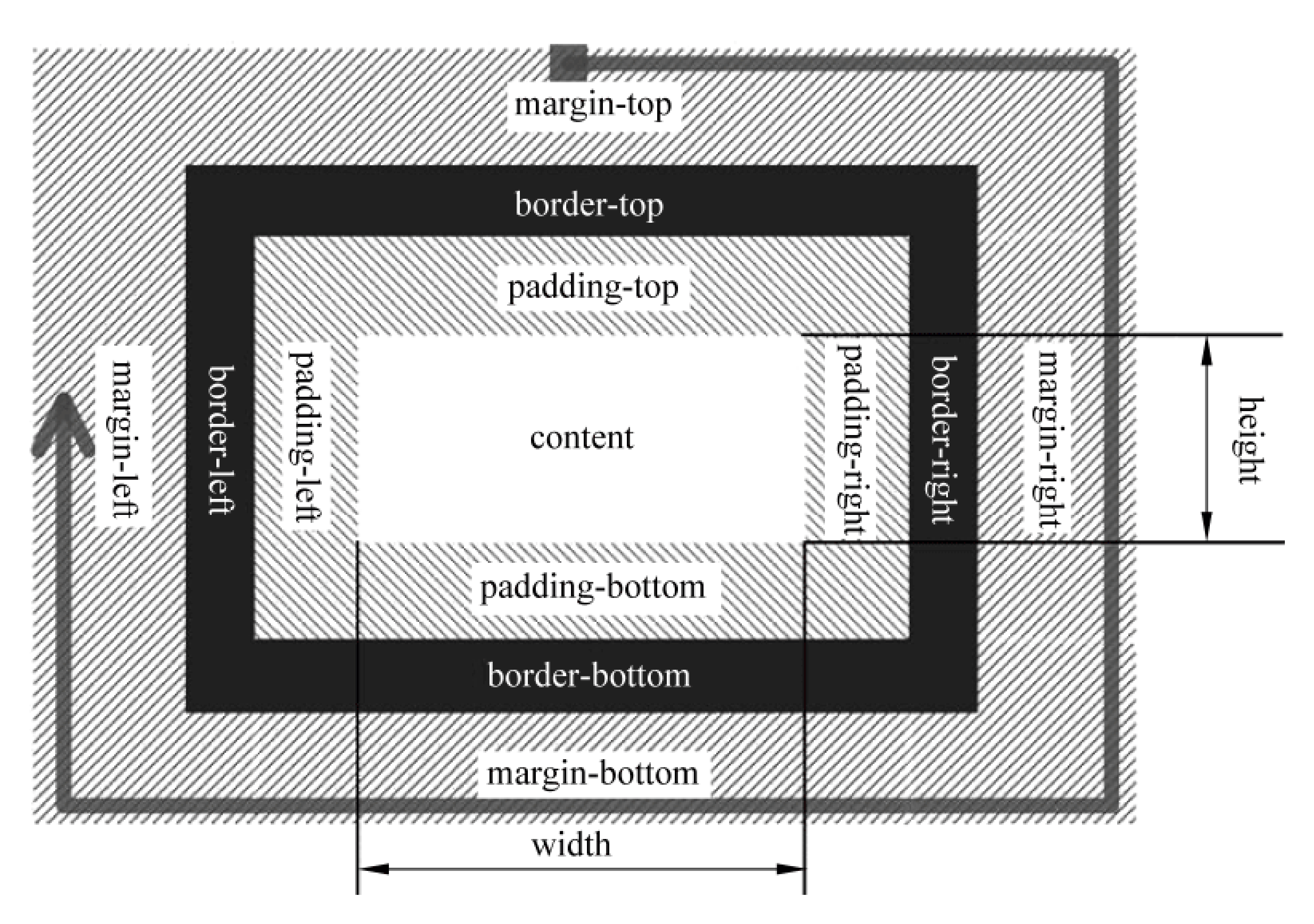


图 4-22　盒子模型及有关属性

盒子的总尺寸。

因此一个元素盒子的实际宽度如下：

实际宽度＝左边界＋左边框＋左填充＋内容宽度＋右填充＋右边框＋右边界

例如，一个 div 元素的 CSS 样式定义如下：

```
div{
    background: #9cf;
    margin: 20px;
    border: 10px solid #039;
    padding: 40px;
    width: 200px;
    height:88px;     }
<div>盒子模型</div>
```

则该元素占据的网页总宽度是：20＋10＋40＋200＋40＋10＋20＝340(px)。其中，该元素内容占据的宽度是 200px，高度是 88px。

由于默认情况下绝大多数元素的盒子边框宽度是 0，盒子的背景是透明的，所以在不设置 CSS 样式的情况下元素的盒子不可见，但这些盒子依然是占据网页空间的。

通过 CSS 重新定义元素样式，我们可以分别设置元素盒子的 margin、padding 和 border 的宽度值，还可以设置盒子边框和背景的颜色，巧妙设置可以美化网页元素。

2. 边框 border 属性

盒子模型的 margin 和 padding 属性比较简单，只能设置宽度值，最多分别对上、右、下、左分别设置宽度值。而边框 border 则可以设置边框的宽度、颜色和样式。border 属性有 3 个子属性，分别是 border-width(宽度)、border-color(颜色)和 border-style(样式)。在设置 border 时常常需要将这 3 个属性结合起来才能达到良好的效果。

这里重点讲解 border-style 属性,它可以将边框设置为实线(solid)、虚线(dashed)、点划线(dotted)、双线(double)等,效果如图 4-23 所示。

各种样式边框的显示效果在 IE 和 Firefox 中略有区别。对于 groove、inset、outset 和 ridge 这 4 种值,IE 都不支持。下面是图 4-23 对应的代码。

图 4-23 边框样式(border-style 属性)的不同取值的效果

```
<style type="text/css">
div {
    border:6px black;
    margin:6px;
    padding:6px;     }
</style>
<div style="border-style:solid">The border-style of solid.</div>
<div style="border-style:dashed">The border-style of dashed.</div>
<div style="border-style:dotted">The border-style of dotted.</div>
<div style="border-style:double">The border-style of double.</div>
```

我们还可以分别对某个边框设置样式,下面是一些例子,其显示效果如图 4-24 所示。

```
border: 4px solid red;              /* 同时设置 4个边框 */
    /*--------------------------------*/
border-bottom: 6px double black;    /* 单独设置下边框为黑色双线 */
    /*--------------------------------*/
border:3px dotted #00f;
border-right:none;                  /* 设置右边无边框,其他边框为虚线 */
    /*--------------------------------*/
border:5px dashed #666;
border-width:0 5px;                 /* 设置上下无边框 */
```


图 4-24 边框样式的设置效果

提示:当有多条规则作用于同一个边框时,则后面设置的样式会覆盖前面的设置。

实际上,边框 border 属性有个有趣的特点,即两条交会的边框之间是一个斜角,我们可以通过为边框设置不同的颜色,再利用这个斜角,制作出像三角形一样的效果。例如图 4-25,第一个元素将四条边框设置为不同的颜色,并设置为 10 像素宽,此时可明显地看到边框交会处的斜角;第二个元素在第一个元素基础上将元素的内容设置为空,这时由于没有内容,四条边框紧挨在一起,形成四个三角形的效果。第三个元素和第四个元素的内容也为空,第三个元素将左边框设置为白色,下边框设置为红色(当然也可设置上边框

为白色，右边框为红色，效果一样）。第四个元素将左右边框设置为白色，下边框设置为红色，并且左右边框宽度是下边框的一半。



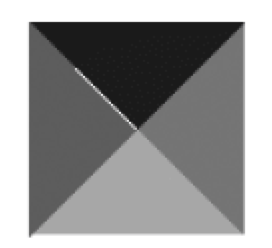
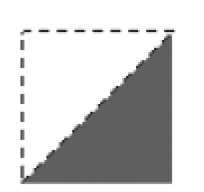
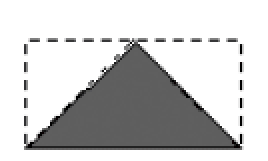




图 4-25　四个元素的边框样式

用代码实现这些效果时，还必须将元素设置为以块级元素显示等，这些在 4.7.5 节制作缺角的导航条中再详细讨论。

3. 填充 padding 属性

填充 padding 属性也称为盒子的内边距。位于盒子的边框和内容之间，和表格的填充属性（cellpadding）相似。如果填充属性为 0，则盒子的边框会紧挨着内容，这样通常不美观。

当对盒子设置了背景颜色或背景图像后，那么背景会覆盖 padding 和内容组成的区域，并且默认情况下背景图像是以 padding 的左上角为基准点在盒子中平铺的，如图 4-26 所示。



图 4-26　背景图像覆盖 padding 区域

4. 边界 margin 属性

边界 margin 位于盒子边框的外侧，也称为外边距。其不会应用背景，因此该区域总是透明的。通过设置 margin，可以使盒子与盒子之间留有一定的间距，而使页面不至于过于拥挤。可以统一设置 4 个边界的宽度，也可单独设置各边界的宽。例如：

```
margin:4px 8px;            /* 上下 4px,左右 8px */
margin-left: -10px;        /* 左边界-10px */
```

5. 盒子模型属性缩写技巧

CSS 缩写是指将多条 CSS 属性集合写到一行中的编写方式，通过对盒子模型属性的缩写，可大大减少 CSS 代码，使代码更清晰，主要的缩写方式有：

1）盒子边界、填充或边框宽度的缩写

（1）对于盒子 margin、padding 和 border-width 的宽度值，如果只写一个值，则表示它四周的宽度相等，例如，p{margin：0px}。

如果给出了 2 个、3 个或者 4 个属性值，它们的含义将有所区别，具体含义如下：

（2）如果给出 2 个属性值，那么前者表示上下边距的宽度，后者表示左右边框的宽度；

（3）如果给出 3 个属性值，那么前者表示上边距的宽度，中间的数值表示左右边框的宽度，后者表示下边距的宽度；

(4) 如果给出4个属性值,依次表示上、右、下、左边距的宽度,即按顺时针排序。

2) 边框border属性的缩写

边框border是一个复杂的对象,它可以设置4条边的不同宽度、不同颜色以及不同样式,所以border属性提供的缩写形式也更为丰富。不仅可以对整个属性进行缩写,也可以对单个边进行缩写。对于整个属性的缩写形式如下:

```
border: border-width | border-style | border-color
```

例如,下面的代码将所有div元素的4条边均设置为1px宽、实线、蓝色边框样式。

```
div{border : 1px solid blue;      }
```

如果要为4条边定义不同的样式,则可以缩写如下:

```
p{  border-width:1px 2px 3px 4px;     /* 上  右  下  左 */
    border-color:white blue red;      /* 上  左右  下 */
    border-style: solid dashed;       /* 上下  左右 */
}
```

如果要单独对某一条边的某个属性进行设置,则可以这样写:

```
border-right-color:red;           /* 设置右边框为红色 */
border-top-width:4px;             /* 设置上边框宽度为4像素 */
```

6. 盒子模型其他需注意的问题

关于盒子模型,还有以下几点需要注意:

(1) 边界margin值可为负,如margin: -480px,填充padding值不可为负。

(2) 如果盒子中没有内容(即空元素,如<div></div>),对它设置的宽度或高度为百分比单位(如width:30%),而且没有设置border、padding或margin值,则盒子不会被显示,也不会占据空间,但是如果对空元素的盒子设置的宽或高是像素值的话,盒子会按照指定的像素值大小显示。

(3) 对于IE 6浏览器,如果网页头部没有定义文档类型声明DOCTYPE,或定义不正确(最常见的是DOCTYPE没有写在HTML文档的第一行),那么IE 6将进入怪异(quirk)模式,此时盒子的宽度width或高度height等于原来的宽度或高度再加上填充值和边框值。因此,在使用了盒子模型属性后,一定要有文档类型声明。

7. 各种元素盒子模型属性的浏览器默认值

所谓浏览器的默认样式就是指不设置任何CSS样式的情况下浏览器对元素样式的定义,例如,对于标题元素,浏览器默认会以粗体的形式显示,我们对元素定义CSS样式,实际上就是覆盖浏览器对元素默认的样式定义。各种元素的浏览器的默认样式如下:

(1) 绝大多数html元素的margin、padding和border属性浏览器默认值为0;

(2) 有少数html元素的margin和padding浏览器默认值不为0。主要有body、h1-h6、p、ul、li、form等,因此有时必须重新定义它们的这些属性值为0。

(3) 表单中大部分 input 元素(如文本框、按钮)的边框属性默认不为 0,有时可以对 input 元素边框值进行重新定义,以达到美化表单中文本框和按钮的目的。

4.4.2　盒子模型的应用

学习了盒子模型以后,我们可以为网页中的任何元素添加填充、边框和背景等效果,只要运用得当,能很方便地美化网页。下面以美化表单和制作特殊效果表格来展示盒子模型的运用技巧。

1. 美化表单

网页中的表单控件在默认情况下背景都是灰色的,文本框边框是粗线条带立体感的,不够美观。下列代码(4-1.html)通过 CSS 改变表单的边框样式、颜色和背景颜色让文本框,按钮等变得漂亮些,效果如图 4-27 所示。

```
<style type="text/css">
form{
    border: 1px dotted #999;            /* 设置 form元素边框为点线 */
    padding: 1px 6px;
    margin: 0;                          /* 清除 form元素的默认边界值 */
    font:14px Arial; }
input{
    color: #00008B;                     /* 对所有 input 标记设置统一的文本颜色 */}
input.txt{                              /* 文本框单独设置 */
    background-color: #fee;
    border:none;                        /* 先清除文本框的边框 */
    border-bottom: 1px solid #266980;   /* 设置文本框下边框的样式 */
    color: #1D5061;}
input.btn{                              /* 按钮单独设置 */
    color: #00008B;
    background-color: #ADD8E6;
    border: 1px outset #00008B;
    padding: 3px 2px 1px;
}
select{                                 /* 设置下拉框样式 */
    width: 100px;
    color: #00008B;
    background-color: #ADD8E6;
    border: 1px solid #00008B;
}
textarea{                               /* 设置多行文本域样式 */
    width: 200px;
    height: 40px;
    color: #00008B;
    background-color: #ADD8E6;
```

```
        border: 1px inset #00008B;     }
</style>
<form action="" method="post">
  <p>用户名:<input class="txt" type="text" name="comments" size=15/></p>
  <p>密 码 :<input class="txt" name="passwd" type="password" size="15" />
  </p>
  <p>所在地:  <select name="addr">
           <option value="1">湖南</option>
           <option value="2">广东</option>
           <option value="3">江苏</option>
           <option value="4">四川</option></select></p>
    <p>个性签名:<br/><textarea name="sign" cols="20" rows="4"></textarea>
    </p>
  <p><input class="btn" type="submit" value="登 录" />
    <input class="btn" type="reset" value="重 置" /></p>
</form>
```

在上述代码中，使用input.txt选中了类名为txt文本框，对于支持属性选择器的浏览器来说，还可以使用input[type="text"]来选中文本框，这样就不需要添加类名。可以看到，美化表单主要就是重新定义表单元素的边框和背景色等属性，对于Firefox来说，还可以为表单元素定义背景图像(background-image)，但IE不支持。

2. 制作1px带虚线边框表格

通过CSS盒子模型的边框属性可以很容易地制作出如图4-28所示的1px虚线边框的表格。方法是首先在把表格的HTML边框属性设置为0，然后给表格table用CSS添加1px的实线边框，再给第一行的单元格td用CSS添加虚线的下边框。为了让单元格的虚线和边框和表格的边框不交合，设置表格的间距(cellspacing)不为0即可。代码如下：



图4-27 CSS美化表单效果

课程简介

电子商务专业的学生应掌握基本网页设计和制作技能。因为以后要接触到大量的修改网页的工作，至少应该为以后工作打下一个良好的基础。

图4-28 CSS虚线边框表格

```
<style type="text/css">
table {
```

```
    border: 1px solid #03F;    }
td.title {
    border-bottom: 1px dashed #06f;    }
</style>
<table width="168" border="0" cellpadding="3" cellspacing="8">
  <tr><td class="title">课程简介</td></tr>
  <tr><td class="test">电子商务专业……</td></tr>
</table>
```

4.4.3　盒子在标准流下的定位原则

CSS 中有三种基本的定位机制，即标准流(normal flow)、浮动(float)和定位属性下的定位。除非设置了浮动属性或定位属性，否则所有盒子都是在标准流中定位的。

顾名思义，标准流中元素盒子的位置由元素在 HTML 中的位置决定。也就是说，行内元素的盒子在同一行中水平排列；块级元素的盒子占据一整行，从上到下一个接一个排列；盒子可以按照 HTML 元素的嵌套方式包含其子元素的盒子，盒子与盒子之间的距离由 margin 和 padding 决定。在网页中添加一个 HTML 元素也就是向浏览器中插入了一个盒子。

例如，下列代码中有一些行内元素和块级元素，其中块级元素 p 还嵌套在 div 块内。下面采用“*”选择器让网页中所有元素显示“盒子”，效果如图 4-29 所示。

```
<html><head>
<style type="text/css">
 *  {border: 2px dashed #FF0066;
    padding: 6px;
    margin: 2px;}
body{ border: 3px solid blue;}
a{ border: 3px dotted blue;}
</style></head>
<body>
<div>网页的 banner(块级元素)</div>
<a href="#">行内元素 1</a><a href="#">行内 2</a><a href="#">行内 3</a>
<div>这是无名块<p>这是盒子中的盒子</p></div></body></html>
```

在图中，最外面的虚线框是 html 元素的盒子，里面的一个实线框是 body 元素的盒子。在 body 中，包括两个块级元素(div)从上到下排列，和三个行内元素(a)从左到右并列排列，还有一个 p 元素盒子嵌套在 div 盒子中，所有盒子之间的距离由 margin 和 padding 值控制。

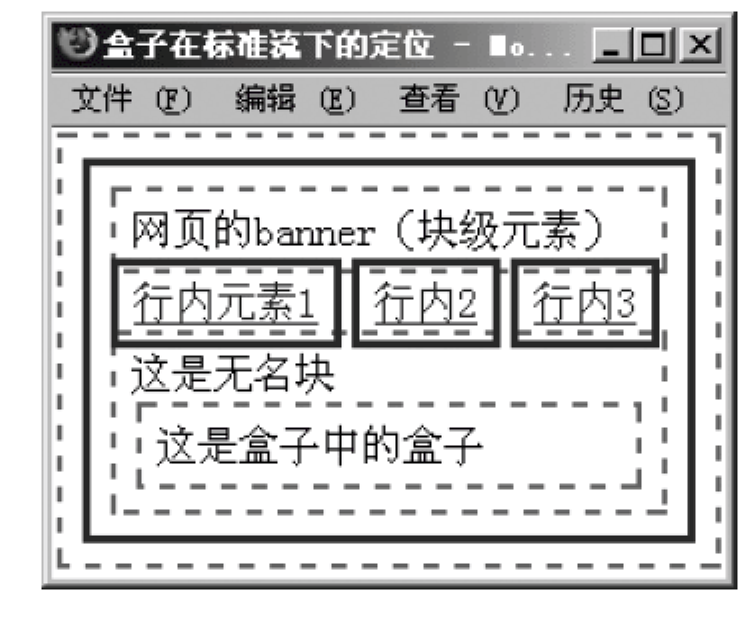


图 4-29　盒子在标准流下的定位

1. 行内元素的盒子

行内元素的盒子永远只能在浏览器中得到一行高度

的空间（行高由 line-height 属性决定，如果没设置该属性，则是内容的默认高度），如果给它设置上下 border、margin、padding 等值，导致其盒子的高度超过行高，那么盒子上下部分将和其他元素的盒子重叠，如图 4-29 所示。

从图 4-30 可以看出，当增加 a 元素的边框和填充值时，行内元素 a 占据的浏览器高度并没有增加，下面这个 div 块仍然在原来的位置，导致行内元素盒子的上下部分和其他元素的盒子发生重叠（此时 a 元素的盒子将叠放在其他盒子上方），而左右部分不会受影响。因此，不推荐对行内元素直接设置盒子属性，一般先设置行内元素以块级元素显示，再对它设置盒子属性。



图 4-30　增大 a 元素的高度后效果

2. display 属性

实际上，标准流中的元素可通过 display 属性来改变元素是以行内元素显示还是以块级元素显示，或不显示。display 属性的常用取值如下：

```
display: block | inline | none | list-item
```

display 设置为 block 表示为以块级元素显示，设置为 inline 表示以行内元素显示，将 display 设置为其他两项的作用如下：

1）隐藏元素（display：none；）

当某个元素被设置成 display：none；之后，浏览器会完全忽略掉这个元素，该元素将不会被显示，也不会占据文档中的位置。像 title 元素默认就是此类型。在制作下拉菜单、Tab 面板时就需要用 display：none 把未激活的菜单或面板隐藏起来。

提示：使用 visibility：hidden 也可以隐藏元素，但元素仍然会占据文档中原来的位置。

2）列表项元素（display：list-item；）

在 html 中只有 li 元素默认是此类型，将元素设置为列表项元素并设置它的列表样式后元素左边将增加列表图标（如小黑点）。

修改元素的 display 属性一般有以下用途：

- 让一个 inline 元素从新行开始（display：block；）；
- 控制 inline 元素的宽度和高度（对导航条特别有用）（display：block；）；
- 无须设定宽度即可为一个块元素设定与文字同宽的背景色（display：inline；）。

3. 上下 margin 合并问题

上下 margin 合并是指当两个块级元素上下排列时，它们之间的边界（margin）将发生合并，也就是说，两个盒子边框之间的距离等于这两个盒子 margin 值的较大者。如图 4-31 所示，浏览器中两个块元素将会由于 margin 合并按右图方式显示。

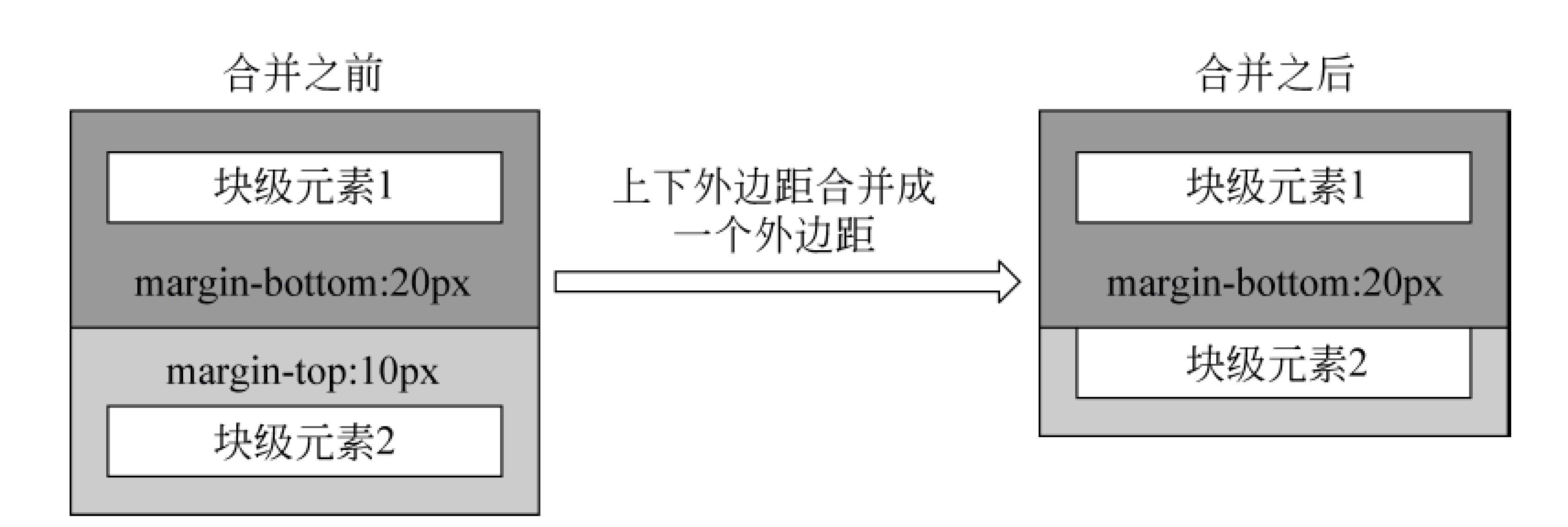


图 4-31　上下 margin 合并

元素上下 margin 合并的一个例子是由几个段落(p 元素)组成的典型文本页面,第一个 p 元素上面的空白等于段落 p 和段落 p 之间的空白宽度。这说明了段落之间的上下 margin 发生了合并,从而使段落各处的距离相等了。

4. 父子元素空白边叠加问题

当一个元素包含在其父元素中时,若父元素的边框和填充为 0,此时父元素和子元素的 margin 挨在一起,那么父元素的上下 margin 会和子元素的上下 margin 发生合并,但是左右 margin 不会发生合并现象,如图 4-32 所示。

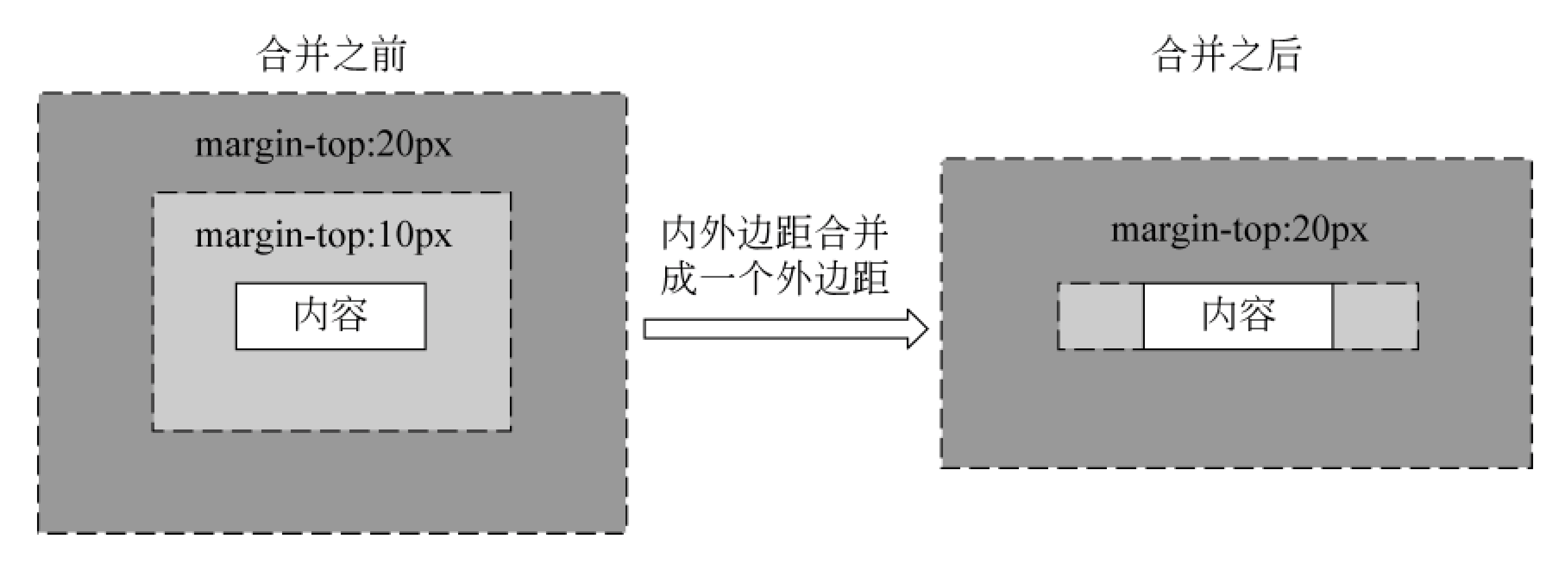


图 4-32　父子元素空白边合并

下面的代码是一个上下 margin 合并的例子,它的显示效果如图 4-33 所示。

```
<style type="text/css">
#inner {
    margin: 30px;
    height: 50px;   width: 200px;
    background-color: #99CCFF;
    border: 1px solid #FF0000;      }
#outer {
    margin: 20px;            /* 父元素只设置了边界,没设置边框和填充 */      }
body { margin: 10px; }
</style>
<body>
    <div id="outer"><div id="inner">此处显示   id "inner" 的内容</div></div>
</body>
```

在图 4-33 中，由于父元素没有设置边框和填充值，使父元素和子元素的上下 margin 发生了合并，而左右 margin 并未合并。如果有多个父元素的边框和填充值都为 0，那么子元素会和多个父元素的上下 margin 发生合并。因此在上例中，上 margin 等于 #inner、#outer、body 三个元素上 margin 的最大值 30px。

若父元素的边框或填充不为 0，或父元素中还有其他内容，那么父元素和子元素的 margin 会被分隔开，因此不存在 margin 合并的问题。

提示：如果有盒子嵌套，要调整外面盒子和里面盒子之间的距离，尽量用外面盒子的 padding 来调整，不要用里面盒子的 margin，以避免父子元素上下 margin 合并现象发生。

5. 左右 margin 不会合并

元素的左右 margin 等于相邻两边的 margin 之和，不会发生合并，如图 4-34 所示。

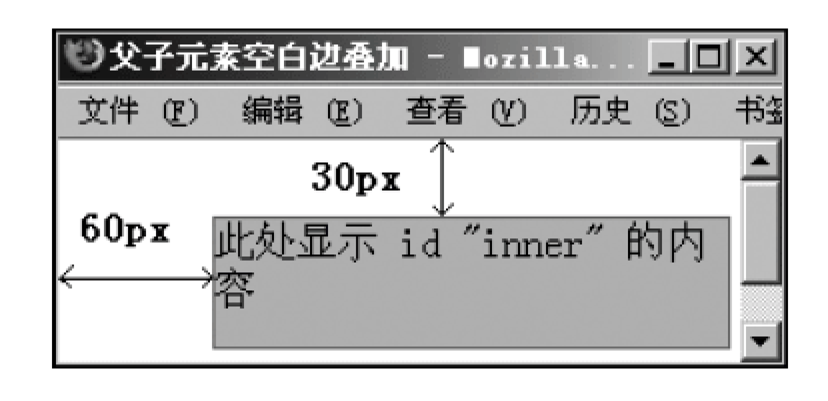


图 4-33 父子元素上下空白边叠加图

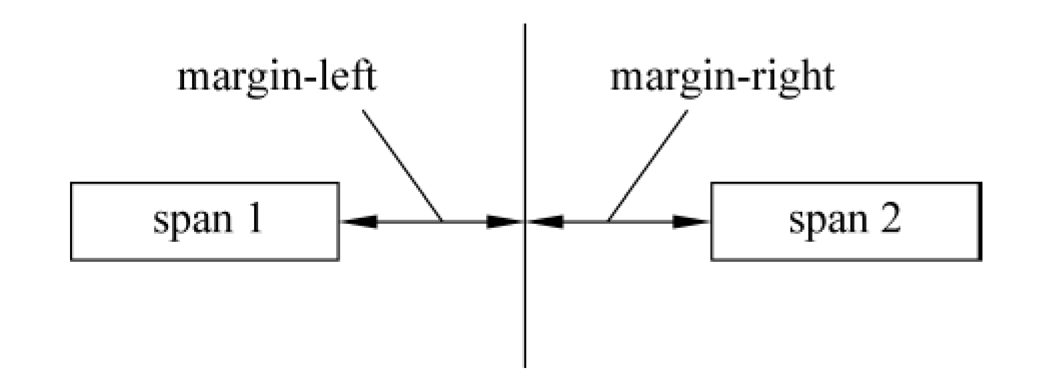


图 4-34 行内元素的左右 margin 不会合并

6. 嵌套盒子在 IE 和 Firefox 中的不同显示

当一个块级元素包含在另一个块级元素中时，若对父块设置高度，但父块的高度不足以容纳子块时，IE 将使父块的高度自动伸展，达到能容纳子块的最小高度为止。而 Firefox 对父块和子块均以定义的高度为准，父块高度不会伸展，任其子块露在外面，子块高度也不会压缩。如图 4-35 所示，其对应的代码如下。



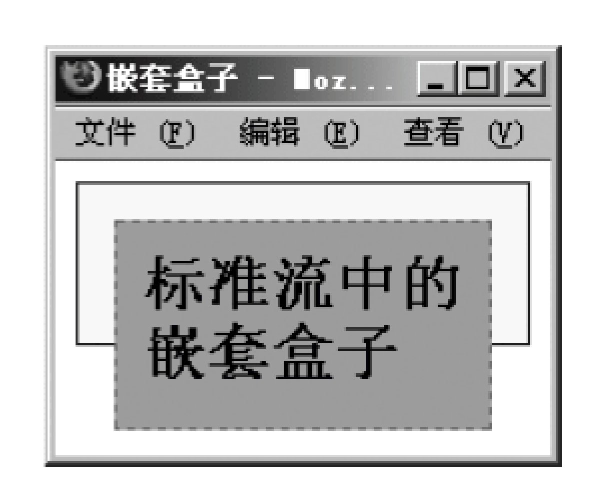


图 4-35 设置父元素和子元素高度后在 IE(左)和 Firefox(右)中的显示效果

```
#outer #inner {
    background-color: #90baff;
    margin: 8px; padding:10px;
    height:60px;
    font-size:24px;   font-weight:bold;
    border: 1px dashed red;      }
#outer {
    border: 1px solid #333;
    padding: 6px;
```

```
    height:50px;
    background-color: #ff9;      }
</style>
<div id="outer">
    <div id="inner">标准流中的嵌套盒子</div>
</div>
```

从这里可以看出，Firefox 对元素的高度解释严格按照我们设定的高度执行，而 IE 对元素高度的设定有点自作主张的味道，它总是使标准流中子元素的盒子包含在父元素盒子当中。从 CSS 标准规范来说，IE 这种处理方式是不符合规范的，它这种方式本应该由 min-height（最小高度）属性来承担。

提示：CSS2 规范中有 4 个相关属性 min-height、max-height、min-width、max-width，分别用于设置最大、最小高度和宽度，IE 6 不支持这 4 个属性，而 IE 7、Firefox 等浏览器都能很好地支持它们。

7. 标准流下定位的应用——制作竖直导航菜单

利用盒子模型及其在标准流中的定位方式，就可以制作出无须表格的竖直菜单，原理是通过将 a 元素设置为块级元素显示，并设置它的宽度，再添加填充、边框和边距等属性实现的。当鼠标滑过时改变它的背景和文字颜色以实现动态交互。代码如下，效果如图 4-36 所示。

```
#nav a {
    font-size: 14px; color: #333;
    text-decoration: none;
    background-color: #ccc;
    display: block;
    width:140px;
    padding: 6px 10px 4px;
    border: 1px solid black;
    margin: 2px;   }
#nav a:hover {
    color: White;
    background-color: #666;      }
<div id="nav">
    <a href="#">首 页</a><a href="#">中心简介</a>
    <a href="#">政策法规</a><a href="#">常用下载</a>
    <a href="#">为您服务</a><a href="#">技术支持和服务</a>
</div>
```

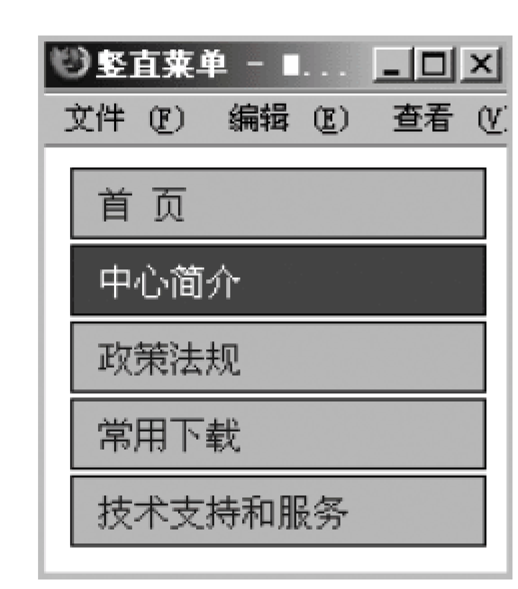


图 4-36　竖直导航菜单

4.5　背景的控制

背景（background）是网页中常用的一种表现方法，无论是背景颜色还是背景图片，只要灵活运用都能为网页带来丰富的视觉效果。

4.5.1 CSS的背景属性

很多HTML元素(如table、td)都具有bgcolor和background等HTML属性，可以用来设置背景颜色和背景图片，但形式比较单一。对背景图片的设定，只支持在X轴和Y轴都平铺的方式。因此，如果同时设置了背景颜色和背景图片，而背景图片又不透明，那么背景颜色将被背景图片完全挡住，只显示背景图片。

而CSS对元素的背景设置，则提供了更灵活的方式，如背景图片既可以平铺也可以不平铺，还可以仅在X轴平铺或仅在Y轴平铺，当背景图片不平铺时，并不会完全挡住背景颜色，因此可以同时设置背景颜色和背景图片，将两者有机地融合在一起。

CSS的背景属性是background，或以“background-”开头，表4-4列出了CSS的背景属性及其可能的取值。

表4-4 CSS的背景属性及其取值

属　性	描　述	可　用　值
background	设置背景的所有控制选项，是其他所有背景属性的缩写	其他背景属性可用值的集合
background-color	设置背景颜色	命名颜色、十六进制颜色等
background-image	设置背景图片	url(URL)
background-repeat	设置背景图片的平铺方式	repeat、repeat-x、repeat-y、no-repeat
background-attachment	设置背景图片固定还是随内容滚动	scroll、fixed
background-position	设置背景图片显示的起始位置(第一个值为水平位置，第二个值为竖直位置)	[left | center | right] [top | center | bottom]或[x%] [y%]或[x-pos] [y-pos]

background属性是所有背景属性的缩写形式，5种背景属性的缩写顺序为：

```
background: background-color||background-image||background-repeat||background-attachment||
background-position
```

例如：

```
body {background:silver url(images/bg5.jpg) repeat-x fixed 50%  50%;}
```

可以省略其中一个或多个属性值，如省略，那么该属性将使用浏览器默认值，默认值为：

- background-color：transparent　　/ * 背景颜色透明 * /
- background-image：none　　/ * 无背景图片 * /
- background-repeat：repeat　　/ * 背景默认完全平铺 * /
- background-attachment：scroll　　/ * 随内容滚动 * /
- background-position：0% 0%　　/ * 从左上角开始定位 * /

说明：

(1) background-repeat属性值可设置为不平铺(no-repeat)、水平平铺(repeat-x)、垂

直平铺(repeat-y)和完全平铺(repeat),其中 repeat-x 和 repeat-y 的效果如图 4-37 所示。

(2) background-position(背景定位)属性值单位中百分数和像素的意义不同,使用百分数定位时,是将背景图片的百分比位置和元素盒子的百分比位置对齐。例如:

```
<div style="width:100px;height:100px;background:url(hua.gif) no-repeat 50% 33%;"></div>
```

就表示将背景图片的水平 50%处和 div 盒子的水平 50%处对齐,竖直方向 33%处和盒子的竖直方向 33%处对齐。这样背景图片将位于盒子的水平中央(相当于设置为 center),垂直方向约 1/3 处。而如果设置为像素,则表示相对于盒子的左边缘或上边缘(边框内侧)偏移指定的距离。图 4-38 对这两种属性值单位进行了对比。

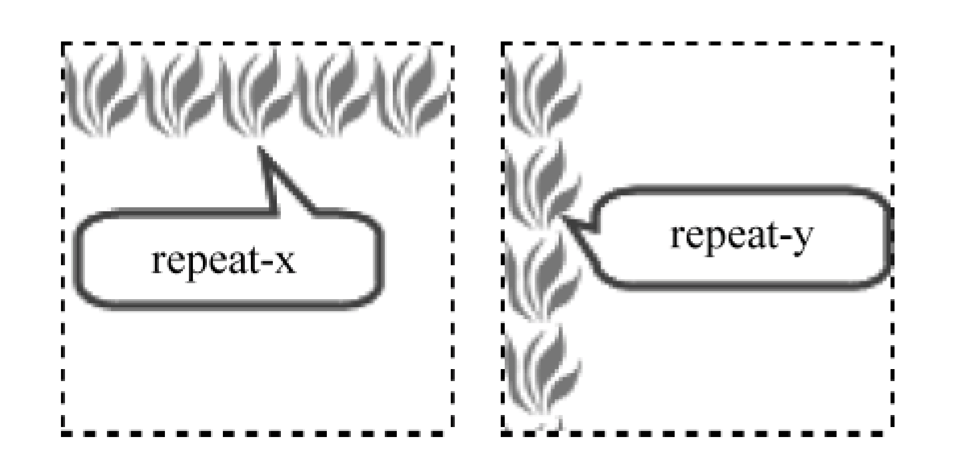


图 4-37 背景水平平铺和垂直平铺的效果

33%
50%
33px
50px

图 4-38 背景定位属性取值单位不同的效果

background-position 的取值还可设置为负数,当背景图像比盒子还大时,设置为负数可以让盒子不显示背景图像的左边部分或上边部分的图案。

背景的所有这些属性都可以在 Dreamweaver 的 CSS 面板的“背景”选项面板中设置,它们之间的对应关系如图 4-39 所示。

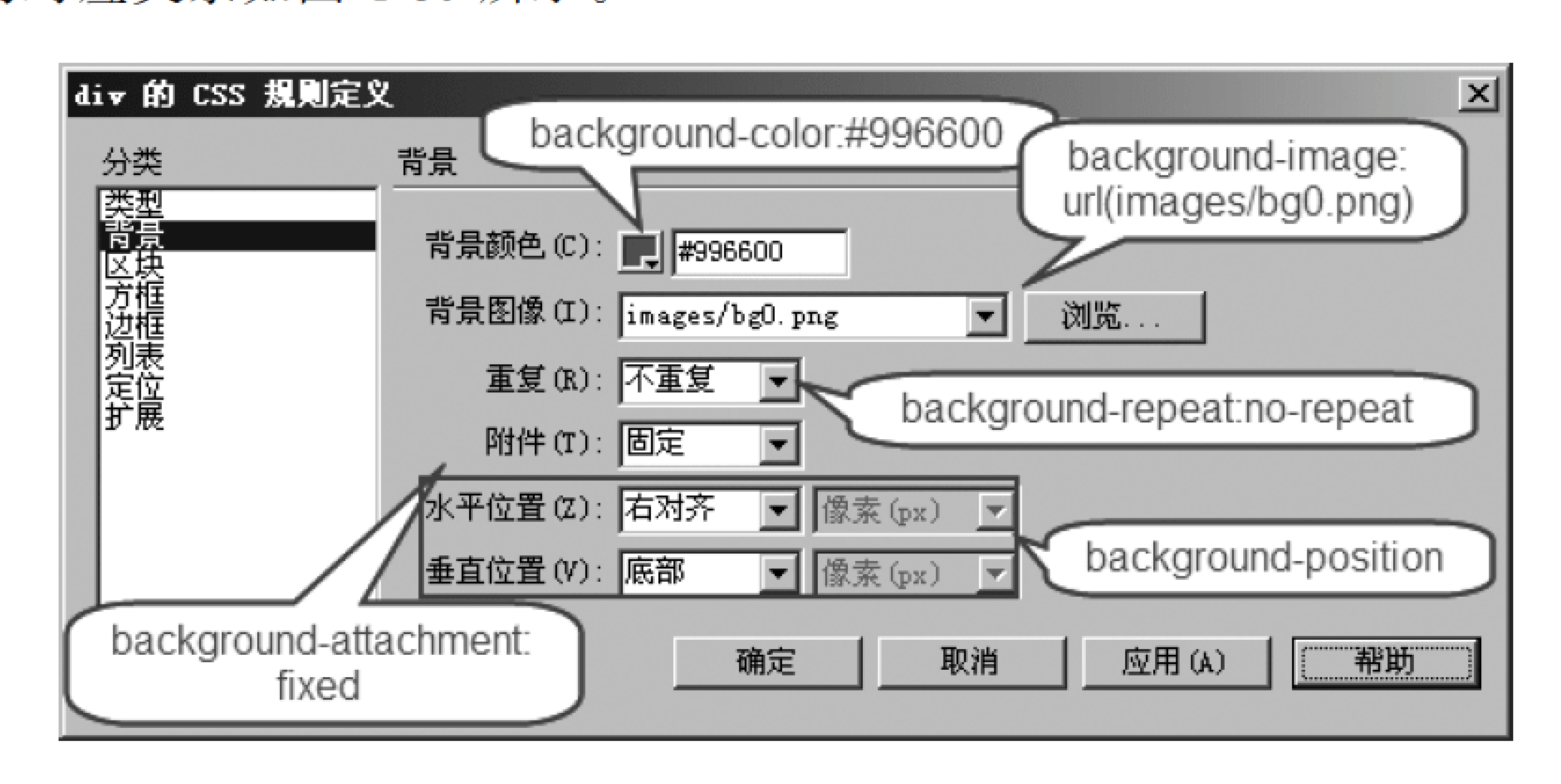


图 4-39 Dreamweaver 中的背景设置面板

4.5.2 背景的基本运用技术

1. 同时运用背景颜色和背景图片

在一些网页中,网页的背景从上到下由深颜色逐渐过渡到浅颜色,由于网页的高度通常不好估计,所以无法只用一幅背景图片来实现这种渐变背景。这时可以对 body 元素同时设置背景颜色和背景图片,在网页的上部采用类似图 4-40 这样很窄的渐变图片水平平铺作为上方的背景,再用一种和图片底部颜色相同的颜色作为网页背景色,这样就实现

了很自然的渐变效果，而且无论页面有多高。

制作的方法是在 CSS 中设置 body 标记的背景颜色和背景图片，并把背景图片设置为横向平铺就可以实现渐变背景了。CSS 代码如下：

```
body{ background:#666 url(images/body_bg.gif) repeat-x; }
```

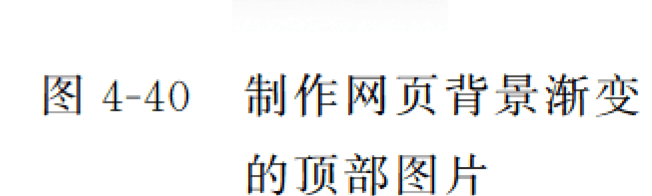


图 4-40　制作网页背景渐变的顶部图片

2. 控制背景在盒子中的位置及是否平铺

在 HTML 中，背景图像只能平铺。而在 CSS 中，背景图像能做到精确定位，允许不平铺，这时效果就像普通的 img 元素一样。例如图 4-41 网页中的茶杯图像就是用让背景图片不平铺并且定位于右下角实现的。实现的代码如下：

```
body { background: #F7F2DF url(cha.jpg) no-repeat right bottom; }
```

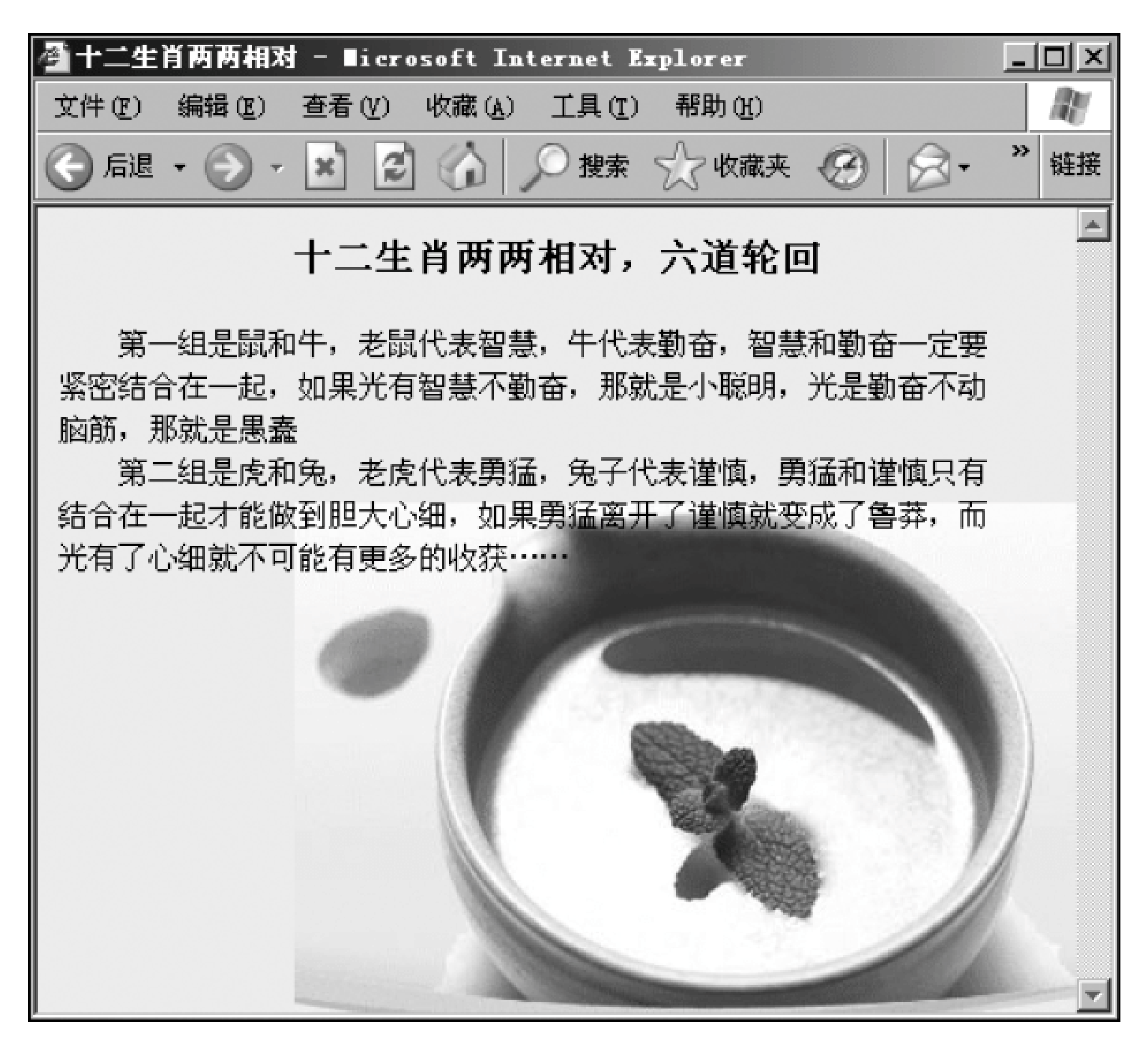


图 4-41　背景图片定位在右下角且不平铺

如果希望图 4-41 中的背景图片始终位于浏览器的右下角，不会随网页的滚动而滚动，则可将 background-attachment 属性设置为 fixed，代码如下：

```
body { background: #f7f2df url(cha.jpg) no-repeat fixed right bottom; }
```

利用背景图像不平铺的方法还可以用来改变列表的项目符号。虽然使用列表元素 ul 的 CSS 属性 list-style-image:url(arrow.gif)可以将列表项前面的小黑点改变成自定义的小图片，但无法调整小图片和列表文字之间的距离。

要解决这个问题，可以将小图片设置成 li 元素的背景，不平铺，且居左，为防止文字遮住图片，将 li 元素的左 padding 设置成 20px，这样就可通过调整左 padding 的值实现精

确调整列表小图片和文字之间的距离了，代码如下，效果如图 4-42 所示。

```
ul{
    list-style-type:none; }
li{
background:url(arrow.gif) no-repeat 0px 3px;     /* 距左边 0px,距上边 3px */
padding-left:20px;     }
```

有了背景的精确定位能力，完全可以使列表项的符号出现在 li 元素中的任意位置上。

→ 中心简介
→ 政策法规汇总
→ 为您服务

图 4-42　用图片自定义项目符号

3. 多个元素背景的叠加

背景图片的叠加是很重要的 CSS 技术。当两个元素是嵌套关系时，那么里面元素盒子的背景将覆盖在外面元素盒子背景之上，利用这一点，再结合对背景图片位置的控制，可以将几个元素的背景图像巧妙地叠加起来。下面以 4 图像可变宽度圆角栏目框的制作来介绍多个元素背景叠加的技巧。

制作可变宽度的圆角栏目框需要 4 个圆角图片，当圆角框制作好之后，无论怎样改变栏目框的高度或宽度，圆角框都能根据内容自动适应。

由于需要四个圆角图片做可变宽度的圆角栏目框，而一个元素的盒子只能放一张背景图片，所以必须准备四个盒子把这四张圆角图片分别作为它们的背景，考虑到栏目框内容的语义问题，这里选择 div、h3、p、span 四个元素，按照图 4-43 的方式设置这四个元素的背景图片摆放位置，并且都不平铺。然后再把这四个盒子以适当的方式叠放在一起。

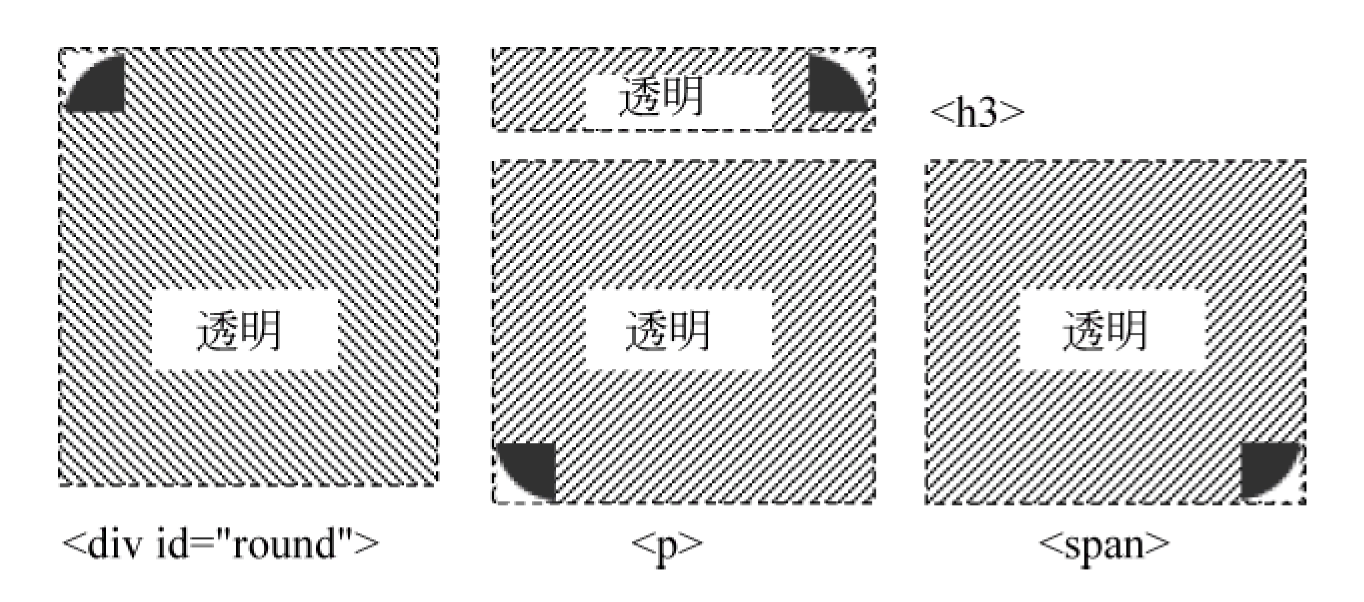


图 4-43　4 图像可变宽度圆角栏目框中 4 个元素盒子的背景设置

从图 4-43 中可以看出，要形成圆角栏目框，首先要把 span 元素放到 p 元素里面，这样它们两个的背景就叠加在一起，形成了下面的两个圆角，然后再把 h3 元素和 p 元素都放到 div 元素中去，就形成了一个圆角框的四个圆角了。因此，结构代码如下：

```
<div id="round">
    <h3>圆角栏目框的标题</h3>
    <p><span>栏目框的内容……</span></p>
</div>
```

由于几层背景的叠加，背景色只能放在最底层的盒子上，因此只能对最外层的 div 元素设置背景色，否则上面元素的背景色会把下面元素的背景图片(圆角)覆盖掉。与此相

反，为了让内容能放在距圆角框边缘有一定内边距的区域，必须设置 padding 值，而且 padding 值只能设置在最里层的盒子(span 和 h3)上。因为如果将 padding 设置在外层盒子(如 p)上，则内外层盒子的边缘无法对齐，就会出现如图 4-44 所示的错误。

接下来对这 4 个元素设置 CSS 属性，主要是将这 4 个圆角图片定位到相应的位置上，span 元素必须设置为块级元素显示，应用盒子属性才会有正确效果。CSS 代码如下：

```
<style type="text/css">
#round{
        font: 12px/1.6 arial;
        background: #abc276 url(images/right-top.gif) no-repeat right top;   }
    #rounded h3 {
        background: url(images/left-top.gif) no-repeat;
        padding: 15px 20px 0;
        color: #fff;                        /* 设置标题的文字颜色为白色 */
        margin: 0;    }
    #rounded p {
        margin: 0;                          /* 清除 p元素的默认边界 */
        text-indent:2em;                    /* 内容部分段前空两格 */
        background: url(images/left-bottom.gif) no-repeat left bottom;     }
    #rounded span{
          padding: 10px 20px 13px;
          display:block;
          background:url(images/right-bottom.gif) no-repeat right bottom;      }
  </style>
```

最终效果如图 4-45 所示。但这个圆角框没有边框，要制作带有边框的可变宽度圆角框，则至少需要 4 张图片通过滑动门技术实现。



图 4-44 错误的背景图像位置

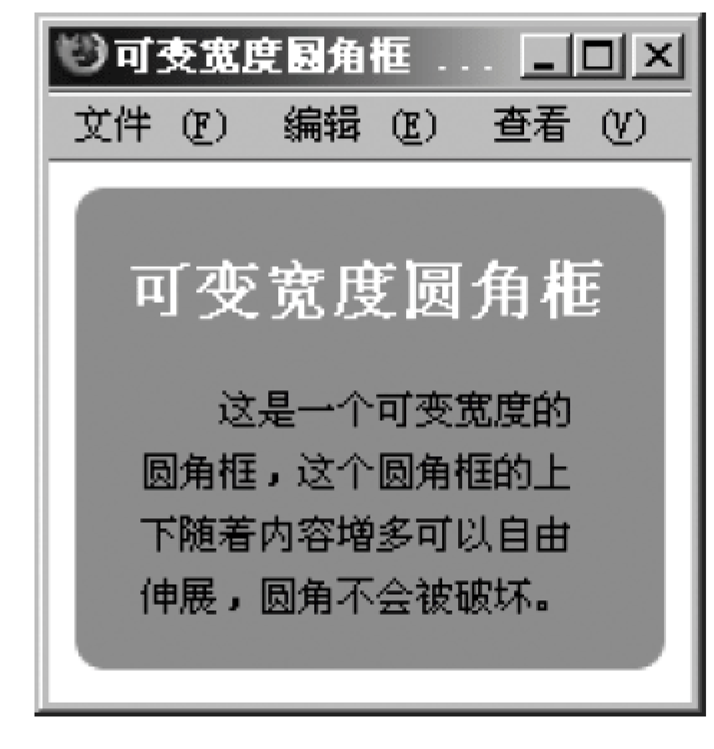


图 4-45 最终的效果

4.5.3 滑动门技术

CSS 中有一种著名的技术叫滑动门技术(sliding doors technique)，它是指一个图像在另一个图像上滑动，将它的一部分隐藏起来，因此而得名。实际上它是一种背景的高级

运用技巧，主要是通过两个盒子背景的重叠和控制背景图片的定位实现的。

滑动门技术的典型应用有：

（1）制作图像阴影；

（2）制作自适应宽度的圆角导航条。

1．图像阴影

阴影是一种很流行、很有吸引力的图像处理技巧，它给平淡的设计增加了深度，形成立体感。使用图像处理软件很容易给图像增添阴影。但是，可以使用 CSS 产生简单阴影效果，而不需要修改底层的图像。通过滑动门技术制作的阴影能自适应图像的大小，即不管图像是大是小都能为它添加阴影效果。这对于交友类网站很适合，因为网友上传的个人生活照片大小一般都是不一样的，而这种方法能自适应地为这些照片添加阴影。

图 4-46 展示了图像阴影的制作过程，在图 4-46 中有 6 张小图，对其进行了编号（①～⑥），在下面的制作步骤中为了叙述方便我们用图①～⑥表示图 4-46 中的 6 张小图。

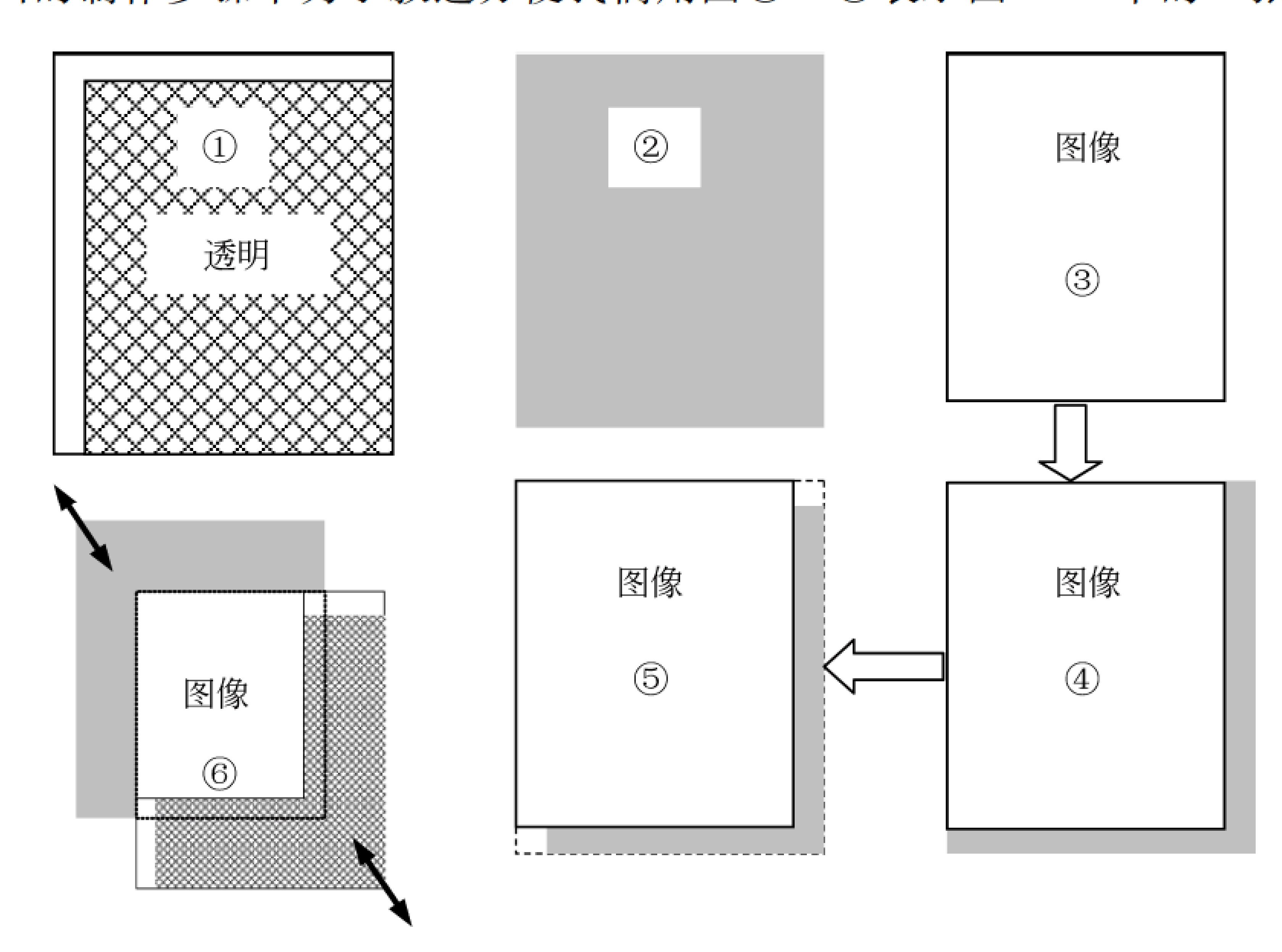


图 4-46　滑动门制作图片阴影原理图

（1）准备一张图①所示的 gif 图片，该图片左边和上边是白色部分，其他区域是完全透明的，将其称为“左上边图片”，然后再准备一张图②所示的灰色图片做背景，灰色图片的右边和下边最好有柔边阴影效果，这两张图片都可以比待添加阴影的图像尺寸大得多。

（2）把待添加阴影的图片③放到灰色图片上面，通过设置图像框的填充值使图像的右边和下边能留出一些，显示灰色的背景，如图④所示，灰色背景图片多余的部分就显示不下了。

（3）接着再把图①的图片插入到图像和灰色背景图片之间，使图①的图片和图像③的图片从左上角开始对齐。这样它的右上角和左下角就挡住阴影了。就出现了如图⑤所示的阴影效果。

（4）图①的图片比图像大一些也没关系，因为图①的图片和图像是左上角对齐的，所以其超出图像盒子的右边和下边部分就显示不下了。而图②的灰色背景图片由于是从右下角开始铺，所以超出图像盒子的左边和上边部分就显示不下了。如图⑥所示，这样图像阴影就能自适应图像大小，就好像①和②两张图片分别向右下和左上两个方向滑动一样。

也可以不用图②的图片文件做灰色的背景，而是直接将img元素的背景设为灰色，再设置它的背景图片为图①的图片，由于背景图片会位于背景颜色上方，这样就出现了没有柔边的阴影效果。代码如下，效果如图4-47所示。

```
img {
    background-color: #CCC;                  /* 灰色背景作为阴影 */
    padding:0 6px 6px 0;                      /* 使右边和下边留出一部分显示灰色背景 */
    background-image: url(top-left.gif);     /* 背景图像为左上边图片 */
}
<img src="works.jpg"/>
```

当然最好先给图片添加边框和填充，使图片出现相框效果，再对它添加阴影效果，这样更美观。由于阴影必须在img图像的边框外出现，所以在img元素的盒子外必须再套一个盒子。这里选择将img元素放入到一个div元素中。代码如下，效果如图4-48所示。

```
.shadow img {
    background-color: #FFF;                  /* 图像填充区的背景为白色 */
    padding: 6px;
    border: 1px solid #333;                  /* 图像边框为灰色 */     }
.shadow {
    background: #ccc url(top-left.gif);          /* 左上边图像将叠放在灰色背景之上 */
    float: left;                             /* 浮动使 div 宽度不会自动伸展 */
    padding:0 6px 6px 0;      }
<div class="shadow"><img src="works.jpg"/></div>
```



图4-47 利用img的背景色和左上边图片制作阴影效果



图4-48 添加了边框后的阴影效果

由于是用背景色做的阴影，所以没有阴影渐渐变淡的柔边效果，为了实现柔边效果，就不能用背景色做阴影，而还是采用图4-46②中一张右边和下边是柔边阴影的图像做阴影。这样img图像下面就必须有两张图片重叠，最底层放阴影图片（图4-46②），上面一层放左上边图片（图4-46①）。因为每个元素只能设置一张背景图片，而为了放两张背景

图片，就必须有两个盒子。因此必须在 img 元素外套两层 div。

另外，我们知道 png 格式的图片支持 Alpha 透明（即半透明）效果，因此可以将左上边图片（图 4-46①）和灰色背景图像交界处的地方做成半透明的白色，保存为 png 格式后引入，这样阴影就能很自然地从白色过渡到灰色，IE 7 和 Firefox 中均能看到这种阴影过渡的 Alpha 透明效果，但 IE 6 由于不支持 png 的 Alpha 透明（但能显示 png 格式的图片），所以看不到柔边效果。实现的代码如下，效果如图 4-49 所示。

```
.shadow img {
    background- color: White;
    padding: 6px;
    border: 1px solid #333;     }
.shadow div {
    background- image: url(top- left.png);
    padding:0 6px 6px 0;         /* 留出两张背景图片的显示位置 * / }
.shadow {
    background: url(images/bottom- right.gif) right bottom;
    float: left;     }
<div class= "shadow"><div><img src= "works.jpg"/></div></div>
```

这样就实现了图像柔边阴影效果，由于左上边图片和 img 图像是左上角对齐，所以如果左上边图片比 img 图像大，即超过了 div 盒子的大小，那么多出的右下部分将显示不下。同样，阴影背景图像与 img 图像从右下角开始对齐，如果背景图像比盒子大，那么背景图像的左上部分也会自动被裁去。所以，我们可以把这两张图片都做大些，就能自适应地为任何大小的图片添加阴影效果。



图 4-49　通过图像实现了柔边的阴影效果

2. 自适应宽度圆角导航条

现在很多网站都使用了圆角形式的导航条，这种导航条两端是圆角，而且还可以带有背景图案，如果导航条中的每一个导航项是等宽的，那么制作起来很简单，用一张圆角图片作为导航条中所有 a 元素的 background-image 就可以了。

但是有些导航条中的每个导航项并不是等宽的，如图 4-50 所示，这时能否仍用一张圆角图片做所有导航项的背景呢？答案是肯定的，使用滑动门技术就能实现：当导航项中的文字增多时，圆角图片就能够自动伸展（当然这并不是通过对图片进行拉伸实现的，那样会使圆角发生变形）。它的原理是用一张很宽的圆角图片给所有导航项做背景。



图 4-50　自适应宽度的圆角导航条

由于导航项的宽度不固定，而圆角总要位于导航项的两端。这就需要两个元素的盒

子分别放圆角图片的左右部分，而且它们之间要发生重叠，所以选择在 a 元素中嵌入 b 元素，这样就得到了两个嵌套的盒子。制作步骤如下：

（1）首先写结构代码。

```
<div id="nav">
    <a href="#"><b>首 页</b></a>
    <a href="#"><b>中心简介</b></a>
    <a href="#"><b>常用下载</b></a>
    <a href="#"><b>为您服务</b></a>
    <a href="#"><b>技术支持和服务</b></a>
</div>
```

（2）分析：a 元素的盒子放圆角图片的左边部分，这可以通过设置盒子宽度比圆角图片窄，让圆角图片作为背景从左边开始平铺盒子，那么圆角图片的右边部分盒子就容纳不下了，效果如图 4-51①所示。

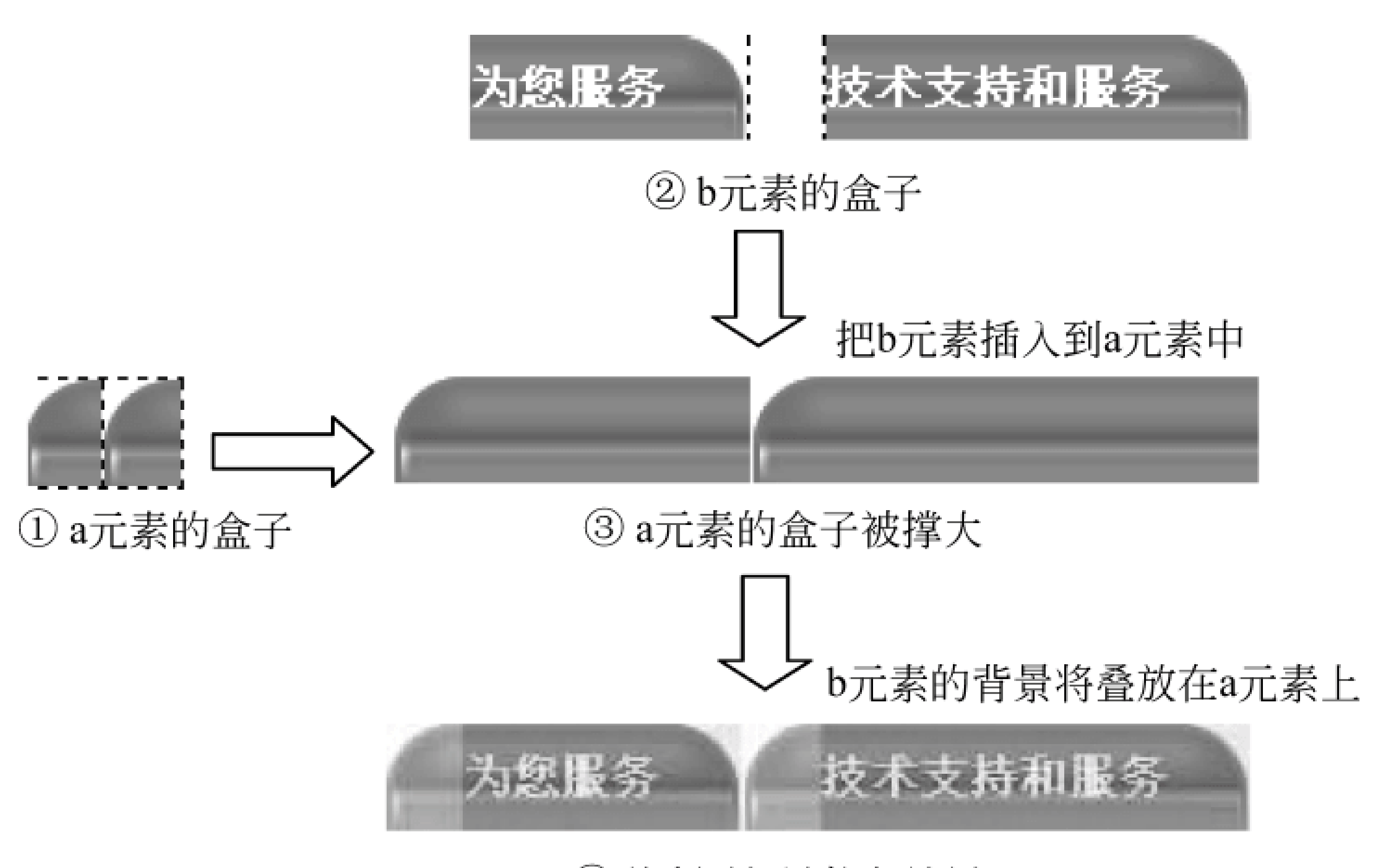


图 4-51 滑动门圆角导航条示意图

b 元素盒子放圆角图片的右边一部分，由于盒子宽度小于圆角图片宽，让圆角图片作为背景从右边开始平铺盒子，那么圆角图片的左边就容纳不下了。效果如图 4-51②所示。

再把 b 元素插入到 a 元素中，这时 a 元素的盒子为了容纳 b 的盒子会被撑大，如图 4-51③所示。这样里面盒子的背景就位于外面盒子背景的上方，通过设置 a 元素的左填充值使 b 的盒子不会挡住 a 盒子左边的圆角，而 b 盒子右边的圆角（上方为不透明白色背景）则挡住了 a 盒子右边的背景，这样左右两边的圆角就都出现了，如图 4-51④所示。同时，改变文字的多少，能使导航条自动伸展，而圆角部分位于 padding 区域，不会影响圆角。

（3）根据以上分析设置外面盒子 a 元素的 CSS 样式：

```
#nav a {
    font-size: 14px; color: white;
```

```
    text-decoration: none;          /* 以上三条为设置文字的一般样式 */
    height: 32px;
    line-height: 32px;              /* 设置盒子高度与行高相等,实现文字垂直居中 */
    padding-left: 24px;             /* 设置左填充为 24px,防止里面的内容挡住左圆角 */
    display: block;
    float: left;                    /* 使导航项水平排列 */
    background: url(round.gif);     /* 背景图像默认从左边开始铺 */}
```

（4）再写里面盒子 b 元素的 CSS 样式代码：

```
#nav a b {
    background: url(round.gif) right top;       /* 使用同一张背景图像但从右边开始铺 */
    display: block;
    padding-right: 24px;                        /* 防止里面的文字内容挡住右圆角 */}
```

（5）最后给导航条添加简单的交互效果：

```
#nav a:hover {
    color: silver;                              /* 改变文字颜色 */ }
```

4.5.4 背景图像的翻转

我们知道,通过背景定位属性(background-position)可以使背景图片从盒子的任意位置上开始显示,如果设置 background-position 为负值,那么将有一部分背景移出盒子,而不会显示在盒子上;如果盒子没有背景那么大,那么只能显示背景图的一部分。

利用这些特点,我们可以将多个背景图像放置在一个大的图片文件里,让每个元素的盒子只显示这张大背景图的一部分,例如制作导航条时,在默认状态下显示背景图的上半部分,鼠标滑过时显示背景图的下半部分,这样就用一张图片实现了导航条背景的翻转。

把多个背景图像放在一个图像文件里的好处有以下两点:

（1）减少了文件的数量,便于网站的维护管理。

（2）鼠标指针移动某个导航项上,如果要更换一个背景图像文件,那么有可能要替换的图像还没有下载下来,就会出现一下停顿,浏览者会不知发生了什么,而如果使用同一个文件,就不会出现这个问题了。

例如,对于自适应宽度圆角导航条来说,我们可以把导航条鼠标离开和滑过两种状态时的背景做在同一个图像文件里,如图 4-52 所示。实现在鼠标滑过时背景图案的翻转,即当鼠标滑过时,让它显示图片的下半部分,默认时则显示图片的上半部分。

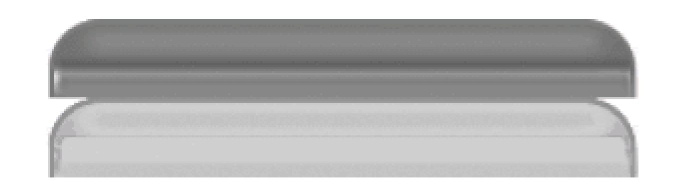


图 4-52 将正常状态和鼠标悬停状态的背景图案放在一张图片 round.gif 中

在前面的自适应宽度圆角导航条的 CSS 代码中添加如下代码。

```
a:hover {
    background-position:0 -32px;               /* 让背景图片从左边开始铺,向上偏移 32px */
```

```
}
a:hover b{
    background-position:100% -32px;                /* 让背景图片从右边开始铺,向上偏移 32px */
    color: red;      }
```

这样,应用了图片翻转的滑动门技术导航条就制作完成了,最终效果如图 4-53 所示。



图 4-53　带有图片翻转效果的滑动门导航条

目前,推荐把许多小的背景图像放在一个图片文件里,这种技术叫作 CSS Sprite 技术。这样可减少要下载的文件数量,从而减少对服务器的请求次数,加快页面载入速度。

4.5.5　CSS 圆角设计

圆角在网页设计中让人又爱又恨,一方面设计师为追求美观的效果经常需要借助于圆角,另一方面为了在网页中设计圆角又不得不增添很多工作量。在用表格设计圆角框时,制作一个固定宽度的圆角框需要一个三行一列的表格,在上下两格放圆角图案。而用表格制作一个可变宽度的圆角框则更复杂,通常采用“九宫格”的思想制作,即利用一个三行三列的表格,把四个角的圆角图案放到表格的左上、右上、左下、右下四个单元格中,把圆角框四条边的图案在表格的上中、左中、右中和下中四个单元格中进行平铺,在中间一个单元格中放内容。而使用 CSS 设计圆角框,则相对简单些,下面对 CSS 圆角设计分类进行讨论。

制作固定宽度的圆角框(不带边框的、带边框的)

用 CSS 制作不带边框的固定宽度圆角框(如图 4-54 所示)至少需要两个盒子:一个盒子放置顶部的圆角图案;另一个盒子放置底部的圆角图案,并使它位于盒子底部。把这两个盒子叠放在一起,再对栏目框设置和圆角相同的背景色就可以了。关键代码如下:

```
#rounded{
        font: 12px/1.6 arial;
        background: #cba276 url(images/bottom.
        gif) no-repeat left bottom;
        width: 280px;
        padding: 0 0 18px;
        margin:0 auto;      }
#rounded h3 {
        background: url(images/top.gif) no-repeat;
        padding: 20px 20px 0;
        font-size: 170%;      color: white;
        line-height:1em;
        margin: 0;      }
<div id="rounded">
    <h3>不带边框的固定宽度圆角框</h3>
```

不带边框的圆角框

这是一个不带边框的固定宽度的圆角框,这个圆角框的上下随着内容增多可以自由伸展,圆角不会被破坏。

图 4-54　不带边框的圆角框

```
    <p>这是一个固定宽度的圆角框……</p>
</div>
```

制作带边框的固定宽度圆角框(如图 4-55 所示)则至少需要三个盒子,最底层的盒子放置圆角框中部的边框和背景组成的图案,并使它垂直平铺,上面两层的盒子分别放置顶部的圆角和底部的圆角,这样在顶部和底部圆角图片就遮盖了中部的图案,形成了完整的圆角框。

带边框的圆角框

这是一个固定宽度的圆角框,由于是固定的宽度,因此制作起来容易且简单。这个圆角框的上下随着内容增多可以自由伸展,圆角不会被破坏。

图 4-55　带边框的圆角框

```
#rounded{
    font: 12px/1.6 arial;
    background: url(images/middle-frame.gif) repeat-y;
    width: 280px;
    padding: 0;
    margin:0 auto;      }
#rounded h3 {
    background: url(images/top-frame.gif) no-repeat;
    padding: 20px 20px 0;
    font-size: 170%;        color: #cba276;
    margin: 0;      }
#rounded p.last {
    padding: 0 20px 18px;
    background: url(images/bottom-frame.gif) no-repeat left bottom;
    height:1%;              /* 防止元素没有内容在 IE 6中不显示 */      }
<div id="rounded">
    <h3>带边框的圆角框</h3>
    <p>这是一个固定宽度的圆角框……</p>
    <p class="last"></p>
</div>
```

需要说明的是,顶部的圆角图案和底部的圆角图案既可以分别做成一张图片,也可以把它们都放在一张图片里,通过控制背景位置来实现显示哪部分圆角。

4.6　盒子的浮动

在标准流中,块级元素的盒子都是上下排列的,行内元素的盒子都是左右排列,如果仅仅按照标准流的方式进行排列,就只有这几种可能性,限制太大。CSS 的制定者也想到了这样排列限制的问题,因此又给出了浮动和定位方式,从而使排版的灵活性大大提高。

例如,如果希望相邻块级元素的盒子左右排列(所有盒子浮动)或者希望一个盒子被另一个盒子中的内容所环绕(一个盒子浮动)做出图文混排的效果,这时最简单的实现办法就是运用浮动(float)属性使盒子在浮动方式下定位。

4.6.1 盒子浮动后的特点

在标准流中，一个块级元素在水平方向会自动伸展，在它的父元素中占满整个一行；而在竖直方向和其他元素依次排列，不能并排，如图 4-56 所示。使用“浮动”方式后，这种排列方式就会发生改变。

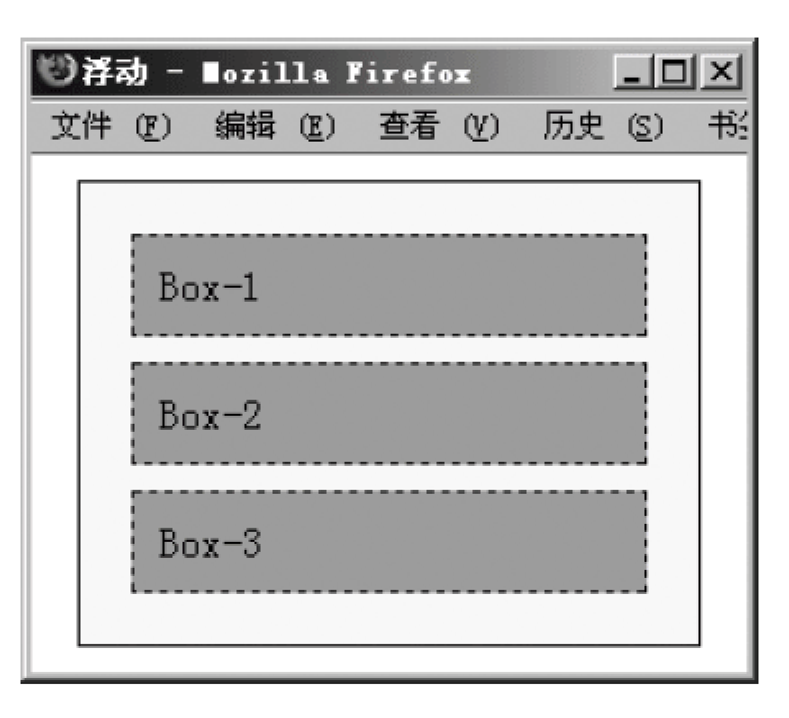


图 4-56 三个盒子在标准流中

CSS 中有一个 float 属性，默认值为 none，也就是标准流通常的情况，如果将 float 属性的值设为 left 或 right，元素就会向其父元素的左侧或右侧靠紧，同时盒子的宽度不再伸展，而是收缩，在没设置宽度时，会根据盒子里面的内容来确定宽度。

下面通过一个实验来演示浮动的作用，基础代码(4-4. html)如下，这个代码中没有使用浮动，它的显示效果如图 4-56 所示。

```
<style type="text/css">
div{
    padding:10px;   margin:10px;
    border:1px dashed #111;
    background-color:#90baff;     }
.father{
    background-color:#ff9;
    border:1px solid #111;     }
</style>
<div class="father">
        <div class="son1">Box-1</div>
        <div class="son2">Box-2</div>
        <div class="son3">Box-3</div>
</div>
```

1. 一个盒子浮动

接下来在上述代码中添加一条 CSS 代码，使 Box-1 盒子浮动。代码如下：

```
.son1{ float:left; }
```

此时显示效果如图 4-57 所示，可发现给 Box-1 添加浮动属性后，Box-1 的宽度不再自动伸展，而且不再占据原来浏览器分配给它的位置。如果再在未浮动的盒子 Box-2 中添一行文本，就会发现 Box-2 中的内容是环绕着浮动盒子的，如图 4-58 所示。

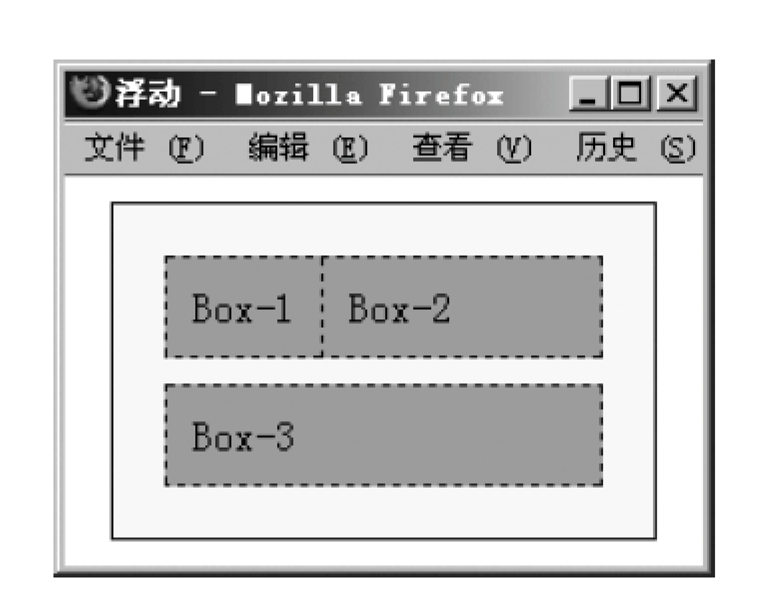


图 4-57　第一个盒子浮动

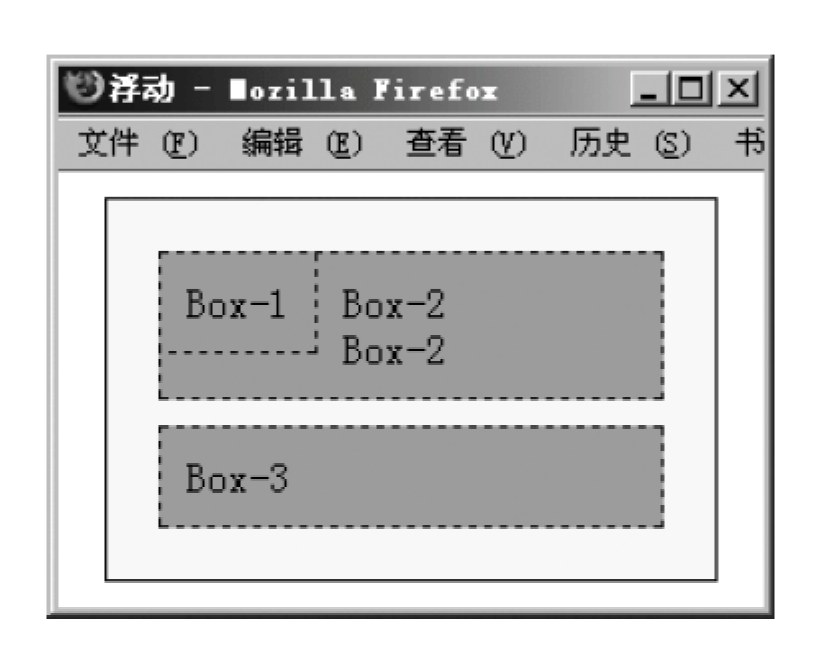


图 4-58　增加第二个盒子的内容

总结：设置元素浮动后，元素发生了如下一些改变：

(1) 浮动后的盒子将以块级元素显示，但宽度不会自动伸展。

(2) 浮动的盒子将脱离标准流，即不再占据浏览器原来分配给它的位置(IE 6 有时例外)。

(3) 未浮动的盒子将占据浮动盒子的位置，同时未浮动盒子内的内容会环绕浮动后的盒子。

提示：所谓“脱离标准流”，是指元素不再占据在标准流下浏览器分配给它的空间，其他元素就好像这个元素不存在一样。例如，在图 4-57 中，当 Box-1 浮动后，Box-2 就顶到了 Box-1 的位置，相当于 Box-2 视 Box-1 不存在一样。但是，浮动元素并没有完全脱离标准流，这表现在浮动盒子会影响未浮动盒子中内容的排列，例如，Box-2 中的内容会跟在 Box-1 盒子之后进行排列，而不会忽略 Box-1 盒子的存在。

2. 多个盒子浮动

在 Box-1 浮动的基础上再设置 Box-2 也左浮动，代码如下：

```
.son2{   float:left;   }
```

此时显示效果如图 4-59 所示(在 Box-3 中添加了一行文本)。可发现 Box-2 盒子浮动后仍然遵循上面浮动的规律，即 Box-2 的宽度也不再自动伸展，而且不再占据原来浏览器分配给它的位置。

如果将 Box-1 的浮动方式改为右浮动，则显示效果如图 4-60 所示，可看到 Box-2 在位置上移动到了 Box-1 的前面。



图 4-59　设置两个盒子浮动



图 4-60　改变浮动方向

接下来再设置 Box-3 也左浮动，此时显示效果如图 4-61 所示。可发现三个盒子都浮

动后，就产生了块级元素水平排列的效果。同时由于都脱离了标准流，导致其父元素中的内容为空。

总结：对于多个盒子浮动，除了每个浮动盒子都遵循单个盒子浮动的规律外，还有以下两条规律：

(1) 多个浮动元素不会相互覆盖，一个浮动元素的外边界(margin)碰到另一个浮动元素的外边界后便停止运动。

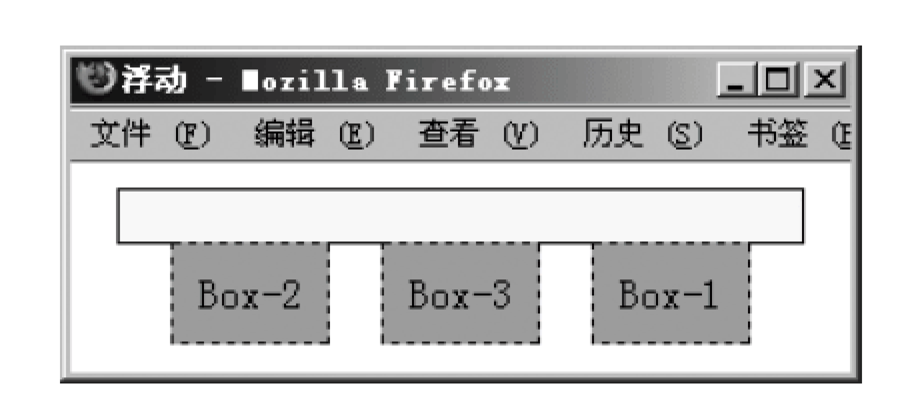


图 4-61　三个盒子都浮动

(2) 若包含的容器太窄，无法容纳水平排列的多个浮动元素，那么最后的浮动盒子会向下移动(见图 4-62)。但如果浮动元素的高度不同，那当它们向下移动时可能会被卡住(见图 4-63)。

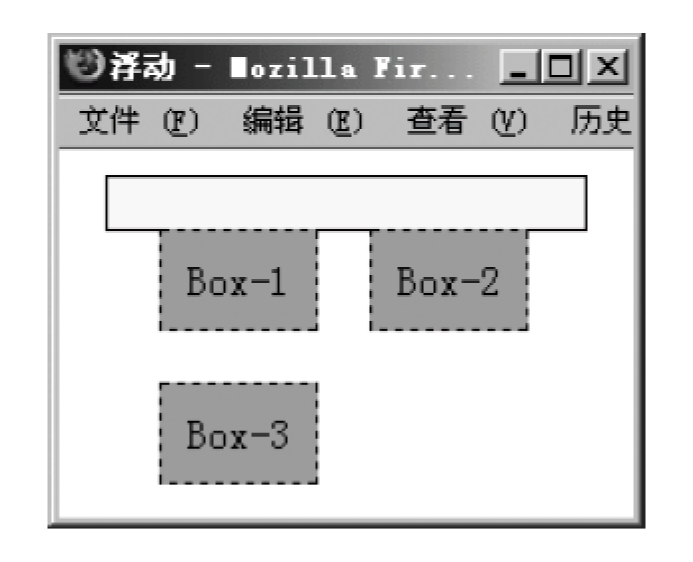


图 4-62　没有足够的水平空间

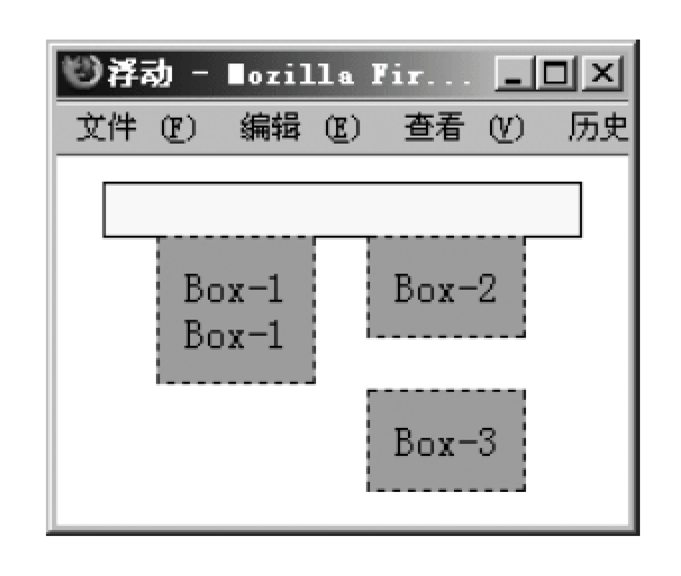


图 4-63　被 Box-1 卡住了

4.6.2　浮动的清除

clear 是清除浮动属性，它的取值有 left、right、both 和 none(默认值)。如果设置盒子的清除浮动属性 clear 值为 left 或 right，则表示该盒子的左边或右边不允许有浮动的对象。值设置为 both，则表示两边都不允许有浮动对象，因此该盒子将会在浏览器中另起一行显示。

例如，在如图 4-60 所示的两个盒子浮动的基础上，设置 Box-3 清除浮动，即在 4-4.html 中添加以下 CSS 代码，效果如图 4-64 所示。

```
.son1{  float:right;  }
.son2{  float:left;   }
.son3{  clear:both;   }
```

可以看到，对 Box-3 清除浮动(clear:both;)，表示 Box-3 的左右两边都不允许有浮动的元素，因此 Box-3 移动到了下一行显示。

实际上，clear 属性既可以用在未浮动的元素上，也可以用在浮动的元素上，如果对 Box-3 同时设置清除浮动和浮动，即：

```
.son3{clear:both; float:left;}
```

则效果如图 4-65 所示，可看到 Box-3 的左右仍然没有了浮动的元素。

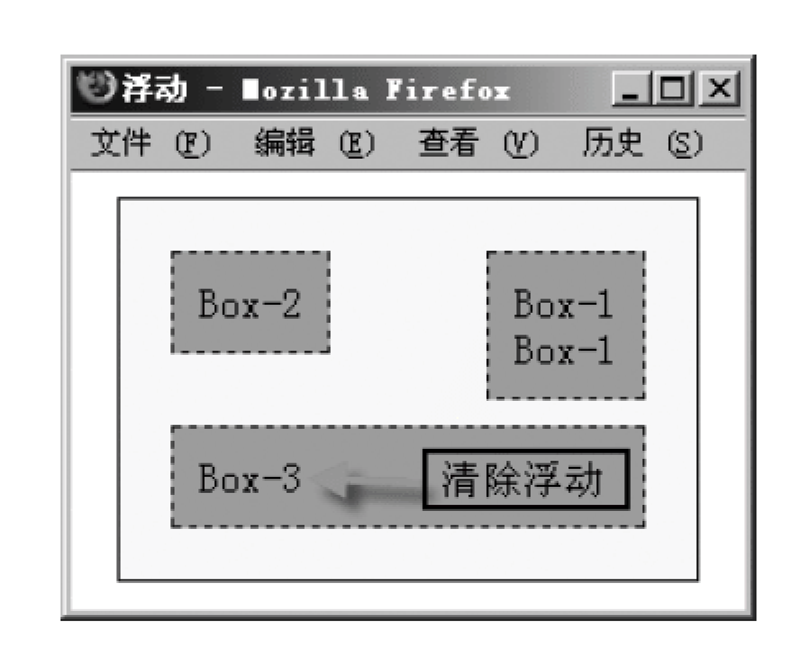


图 4-64　对 Box-3 清除浮动

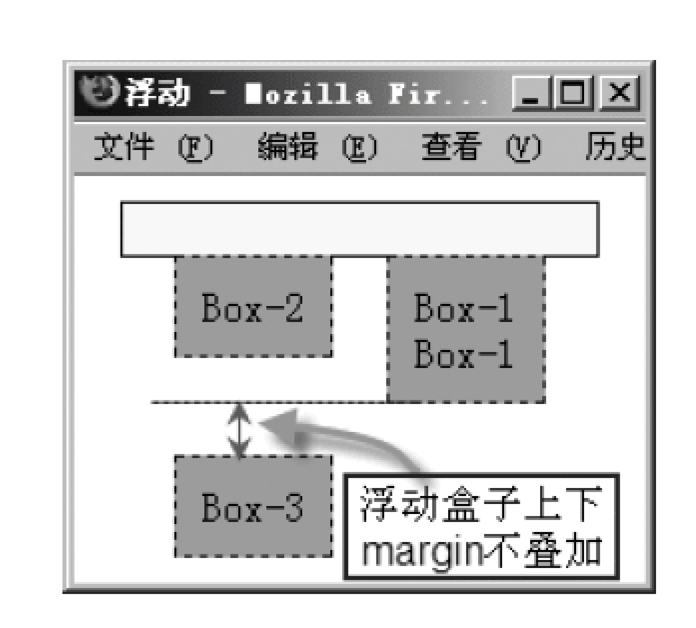


图 4-65　对 Box-3 设置清除浮动和浮动

由此可见，清除浮动是清除其他盒子浮动对该元素的影响，而设置浮动是让元素自身浮动，两者并不矛盾，因此可同时设置元素清除浮动和浮动。

由于上下 margin 叠加只会发生在标准流布局的情况下，而浮动方式下盒子的任何 margin 都不会发生叠加，所以可设置盒子浮动并清除浮动，使上下两个盒子的 margin 不叠加。在图 4-65 中，Box-3 到 Box-1 之间的垂直距离是 20px，即它们的 margin 之和。

提示：在 CSS 布局时，如果发现一个元素移动到了它原来位置的左上方或右上方，并且和其他元素发生了重叠，90%都是因为受到了其他盒子浮动的影响，对其添加一条 clear 属性清除浮动即可。

4.6.3　浮动的浏览器解释问题

设置元素浮动后，浮动元素的父元素或相邻元素在 IE 和 Firefox 中的显示效果经常不一致，这主要是因为浏览器对浮动的解释不同产生的。在标准浏览器中，浮动元素脱离了标准流，因此不占据它原来的位置或外围容器空间。但是在 IE 中（包括 IE 6 和 IE 7），如果一个元素浮动，同时对它的父元素设置宽或高，或对它后面相邻元素设置宽或高，那么浮动元素仍然会占据它在标准流下的空间。下面对这两种情况分别来讨论。

1. 元素浮动但是其父元素不浮动

如果一个元素浮动，但是它的父元素不浮动，那么父元素的显示效果在不同浏览器中可能不同，这取决于父元素是否设置了宽或高。当未设置父元素（外围容器）的宽或高时，IE 和 Firefox 对浮动的显示是相同的，均脱离了标准流。下面是一个示例。

将 4.6.1 节中的结构代码（4-4.html）修改一下，只保留 Box-1，代码如下：

```
<div class="father">
    <div class="son1">Box-1<br />Box-1</div>
</div>
```

然后设置 Box-1 浮动，即“.son1{float:left;}”，此时在 Firefox 和 IE 中的效果如图 4-66 所示。可发现两者效果基本相同。

接下来，我们就对父元素“.father”设置宽度，即添加“.father{width:180px;}”，此时在 Firefox 和 IE 中的效果如图 4-67 所示。可发现在 IE 中浮动元素确实占据了外围容器空间，未脱离标准流。如果对“.father”不设置宽度而是设置高度，也会有类似的效果。





图 4-66　不设置父元素宽度时在 Firefox 和 IE 中的效果



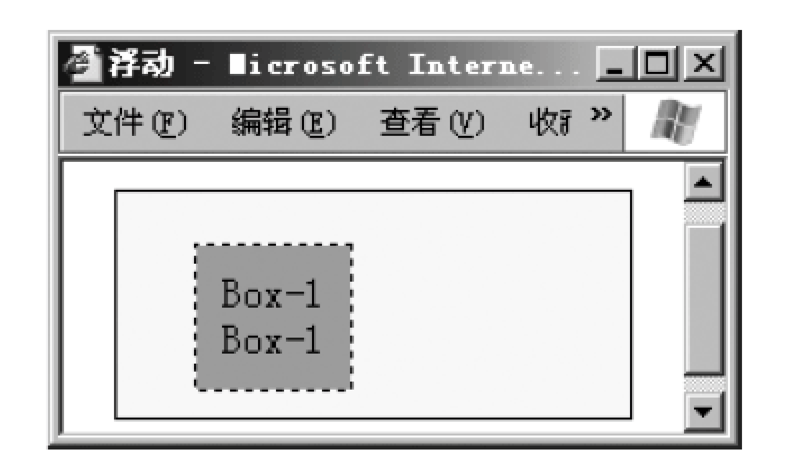


图 4-67　设置父元素宽度时在 Firefox 和 IE 中的效果

可见，当设置了父元素的宽度或高度后，IE 6 中的浮动元素将占据外围容器的空间，而 Firefox 等其他浏览器仍然不占据。

2. 扩展外围盒子的高度(4-5.html)

但是有时我们可能更希望得到图 4-67 中 IE 的这种效果，即让浮动的盒子仍然置于外围容器中。人们把这种需求称为“扩展外围盒子的高度”，要做到这一点，对于 IE 来说只需要设置父元素的宽度或高度就可以了。但对于 Firefox 等标准浏览器，就需要在浮动元素的后面增加一个清除浮动的空元素，来把外围盒子撑开。例如，把上面的结构代码修改如下：

```
<div class="father">
    <div class="son1">Box-1<br />Box-1</div>
<div class="clear"></div>
</div>
```

然后为这个 div 设置样式，CSS 代码如下：

```
.father .clear {
    margin: 0;  padding: 0;  border:0;
    clear: both; }
```

这时在 Firefox 中效果如图 4-68 所示，可看到已经实现了 IE 的这种效果。



图 4-68　对 Firefox 扩展外围盒子高度后

如果不想添加一个空元素，对于支持伪对象选择器的浏览器来说，也可以利用父元素的:after 伪对象在所有浮动子元素后生成一个伪元素，设置这个伪元素为块级显示，清除浮动，也能达到同样的效果。代码如下：

```
div.father:after {
    content:".";            /* 设置伪元素内容,此处可为任意内容 */
    display:block;
    font-size:0; line-height:0; height:0;
                            /* 将伪元素高度等设为 0,使其不占空间 */
    clear:both;             /* 清除浮动 */
    visibility:hidden;      /* 隐藏该伪元素的内容 */
}
```

扩展外围盒子高度的第 3 种方法是设置浮动元素的父元素的 overflow 属性为 hidden,具体请参看 4.7.6 节 overflow 属性用法。

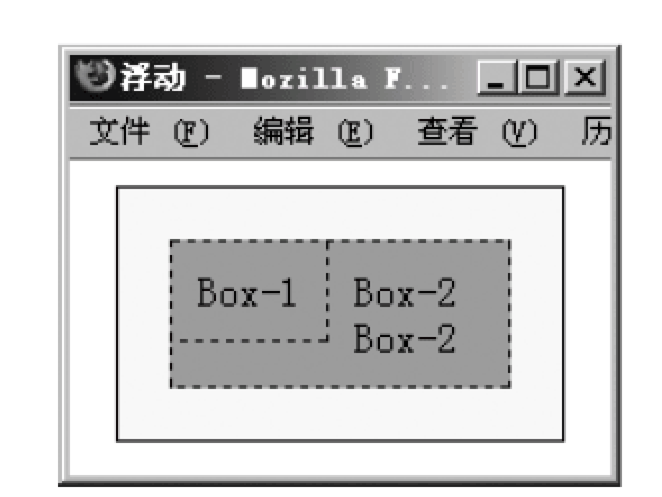
3. 元素浮动但是其后面相邻元素不浮动

如果一个元素浮动,但是它后面相邻的元素不浮动。在不设置后面相邻元素的宽或高时,IE 和 Firefox 显示效果相同。一旦设置了后面相邻元素的宽或高,则在 IE 中,浮动元素将仍然占据它原来的空间,未浮动元素跟在它后面。从本质上来看,这也是 IE 浮动的盒子未脱离标准流的问题。

下面将 4.6.1 节中的结构代码(4-4.html)修改一下,保留 Box-1 和 Box-2,代码如下:

```
<div class="father">
    <div class="son1">Box-1</div>
    <div class="son2">Box-2<br />Box-2<br />Box-2</div>
</div>
```

然后设置“.son1”浮动,即“.son1{float:left;}”,此时在 Firefox 和 IE 6 中的效果如图 4-69 所示。可发现两者效果基本相同。



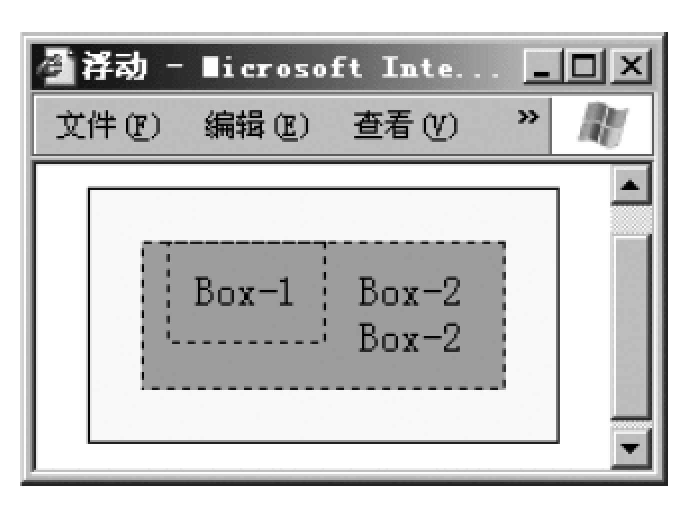


图 4-69 当浮动盒子后面的未浮动盒子未设置宽或高时,Firefox 和 IE 6 显示基本相同

接下来,对“.son2”设置高度,即添加“.son2{height:40px;}”,此时在 Firefox 和 IE 中的效果如图 4-70 所示。可发现在 IE 中浮动元素确实占据了外围容器空间,未脱离标准流。对“.son2”设置宽度也有同样的效果。

提示:为避免出现上述 IE 和 Firefox 显示不一致的情况发生,不要对未浮动盒子设置 width 和 height 值,如果要控制未浮动盒子的宽度,可以对它的外围容器设置宽度。表 4-5 对浮动的浏览器显示问题进行了总结。



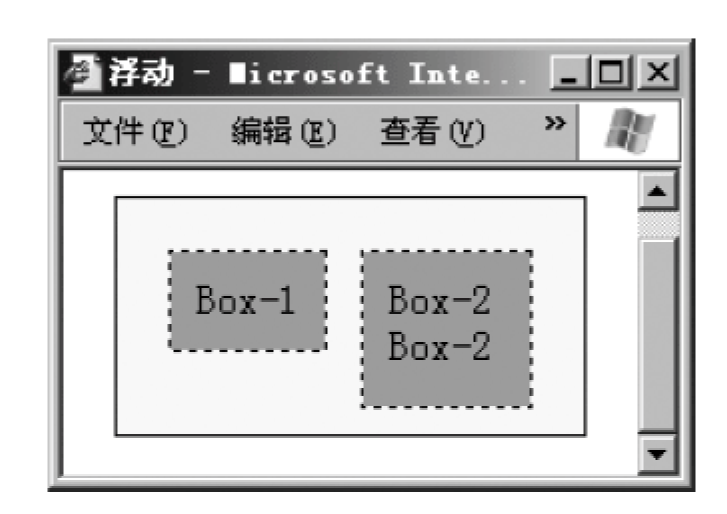


图 4-70　当对未浮动的盒子设置宽或高后，Firefox 和 IE 中显示的差异

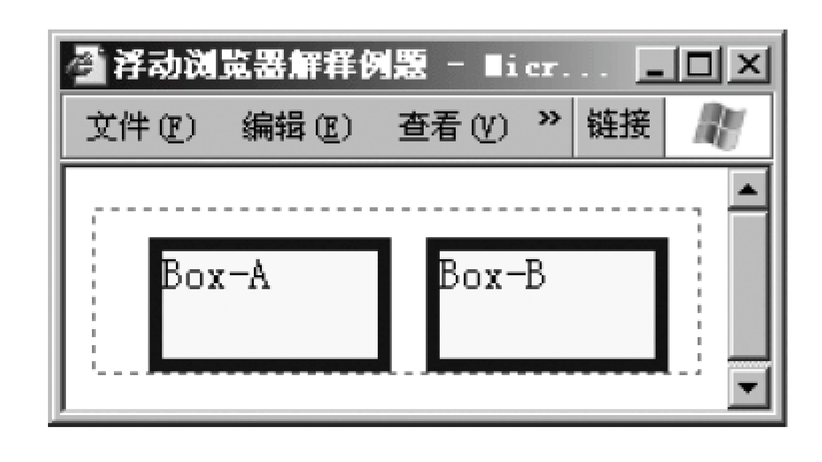
表 4-5　浮动的浏览器显示问题总结

情　况	未浮动的盒子不设置宽或高	对未浮动的盒子设置宽或高
盒子浮动，其外层盒子未浮动	IE 和 Firefox 的显示效果一致	IE 浮动盒子将不会脱离标准流，Firefox 浮动盒子仍然是脱离标准流的
盒子浮动，后面相邻盒子未浮动		

4. 浮动的浏览器解释综合问题

在下面的代码中，第一个盒子浮动，第二个盒子未浮动，而且对两个盒子都设置了高度和宽度，两个盒子大小相等，代码如下，显示效果如图 4-71 所示，请分别解释在 IE 和 Firefox 中为什么会有这样的效果。

```
<style type="text/css">
#a,#b {
    background-color:#ff9;
    margin: 10px;     height: 40px;   width: 80px;
    border: 5px solid #009;}
#a {
    float: left;        }
body {   border: 1px dashed red;   }
</style>
<body>   <div id="a">Box-A</div>
    <div id="b">Box-B</div></body>
```



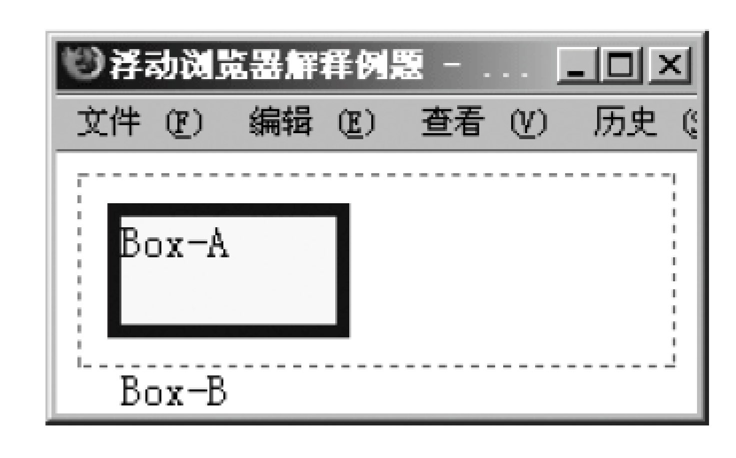


图 4-71　左边是 IE 6 中的显示效果，右边是 Firefox 中的显示效果

答：

(1) 在 IE 6 中，一个盒子浮动，对它后面未浮动的盒子设置了宽或高时，浮动的盒子将不会脱离标准流，仍然占据原来的空间，因为对未浮动的盒子 B 设置了高度和宽度，所

以 A 仍然占据原来的空间，B 就只能排在它后面了。

（2）在 Firefox 中，浮动的盒子总是脱离标准流，不占据空间，所以盒子 B 视 A 不存在，移动到了盒子 A 的位置，由于 A、B 大小相等，盒子 B 正好被盒子 A 挡住。而未浮动盒子的内容将环绕浮动的盒子，所以 B 的内容“Box-B”将环绕盒子 A。由于对盒子 B 设置了宽度，B 的内容“Box-B”只能移到盒子 A 下面去了，B 的内容和盒子 A 之间的距离是盒子 A 的 margin 值。在 Firefox 中，对设置了高度的盒子，盒子高度不会自动伸展，所以盒子 B 的内容就跑到它的外面去了。

如果要使该例中的代码在两个浏览器中显示效果相似，可设置 #b{overflow: hidden;}，通过溢出属性清除盒子 A 浮动对 B 的影响。具体原理请参考 4.7.6 节。

5. IE 6 浮动元素的双倍 margin 错误

在 IE 6 中，只要设置元素浮动，则设置左浮动，盒子的左 margin 会加倍，设置右浮动，盒子的右 margin 会加倍。这是 IE 6 的一个 bug(IE 7 已经修正了这个 bug)。在图 4-69 中 IE 6 与 Firefox 显示效果的差别就是因为这个问题造成的。

由于两个元素的盒子是从 margin 开始对齐的，在 Firefox 中，Box-1 和 Box-2 的 margin 相等，所以它们的左边框也是重合的。而在 IE 6 中，Box-1 由于左浮动导致左 margin 加倍，如图 4-72 所示，所以它的边框就向右偏移了一个 margin 的距离。

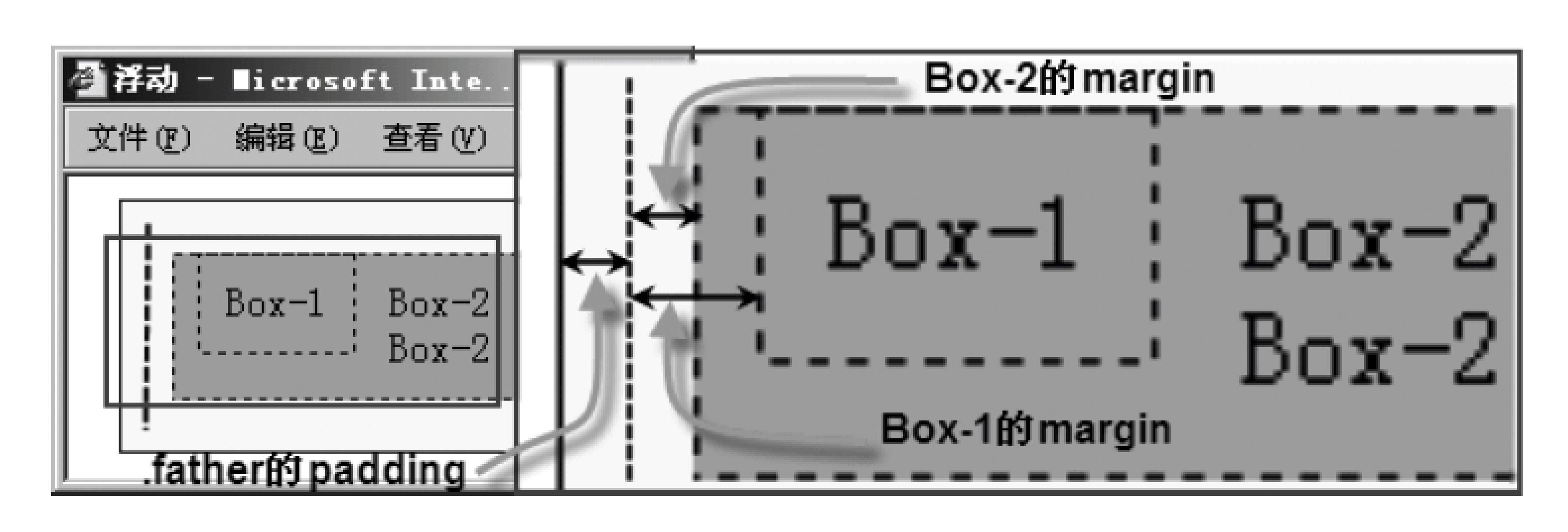


图 4-72　IE 6 双倍 margin 导致的 Box-1 向右偏移

如果将 Box-2 的 margin 重新定义为 0(.son2{margin:0;})，则 Box-2 的边框会向左移一个 margin 的距离，这时可以更清楚地看到 IE 6 中的双倍 margin 错误，在 Firefox 和 IE 6 中的如图 4-73 所示。





图 4-73　IE 6 双倍 margin 错误

解决 IE 6 双倍 margin 错误的方法很简单，只要对浮动元素设置“display:inline;”就可以了。代码如下：

```
.sonl{ float:left;  display:inline; }
```

提示：即使对浮动元素设置“display:inline;”，它仍然会以块级元素显示，因为设置元素浮动后元素总是以块级元素显示的。

当然，也可以不设置浮动盒子的margin，而设置其父元素盒子的padding值来避免这个问题，在实际应用中，可以设置padding的地方尽量用padding，而不要用margin。

4.6.4 浮动的应用举例

1. 图文混排及首字下沉效果等

(1) 如果将一个盒子浮动，另一个盒子不浮动，那么浮动的盒子将被未浮动盒子的内容所包围。如果这个浮动的盒子是图像元素，而未浮动的盒子是一段文本，那么就实现了图文混排效果。代码如下，效果如图4-74所示。



图4-74 图文混排效果

```
<style type="text/css">
img{
    border:1px gray dashed;
    margin:10px 10px 10px 0;
    padding:5px;
    float:left;        /* 设置图像元素浮动 */    }
p{  margin:0;
    font:14px/1.5 "宋体";
    text-indent: 2em;     }
</style>
<img src="images/sheshou.jpg" />
<p>在遥远古希腊的大草原中,驰骋着一批半人半兽的族群,这是一个生性凶猛的族群。"半人半兽"代表着理性与非理性、人性与兽性间的矛盾挣扎,这就是"人马族"。</p>
<p>人马族里唯独的一个例外——奇伦。奇伦虽也是人马族的一员,但生性善良,对待朋友尤以坦率著称,所以奇伦在族里十分受人尊敬</p>
```

(2) 在图文混排的基础上让第一个汉字也浮动，同时变大，则出现了首字下沉的效果，关键代码如下，效果如图4-75所示。

```
.firstLetter{
    font-size:3em;
    float:left;    }
<p><span class="firstLetter">在</span>遥远的古希腊大草原中……。</p>
```

对于首字下沉效果，也可以使用伪对象选择器p:first-letter来选中段落的第一个字符，这样就不需要用<span>标记将段落的第一个汉字括起来了。

（3）如果将第一个段落浮动，则出现了文章导读框效果，代码如下，效果如图 4-76 所示。

```
p{
margin:0;
    font-size:14px;    line-height:1.5;
    text-indent: 2em;    }
.p1{
    width:160px;
    float:left;
    margin:10px 10px 0 0;
    padding:10px;
    border:3px gray double;          /* 双线边框 * /
    background:#9BD;}
<p class="p1">在遥远的古希腊大草原中，驰骋着一批半人半兽的族群，这是一个生性凶猛的族群。
</p>
    <p>"半人半兽"代表着理性与非理性……</p>
```



图 4-75　首字下沉和图文混排效果

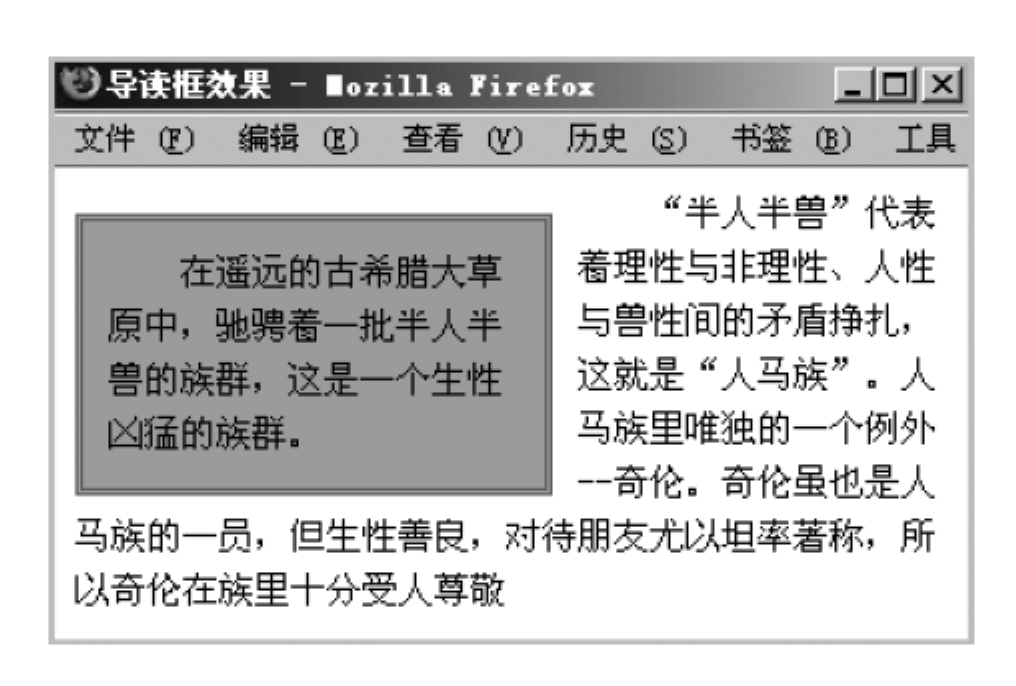


图 4-76　导读框效果

从以上例子可以看出，网页中无论是图像还是文本，对于任何元素，在排版时都应视为一个盒子，而不必在乎元素的内容是什么。

2. 菜单的竖横转换

在 4.4.3 节中，我们利用元素的盒子模型制作了一个竖直导航条。如果要把这个竖直导航条变为水平导航条，只要设置所有 a 元素浮动就可以了，这是因为所有盒子浮动，就能实现水平排列。当然水平导航条一般不需设置宽度，可以把 width 属性去掉。效果如图 4-77 所示。它的结构代码如下：

首 页　中心简介　政策法规　常用下载　为您服务　技术支持和服务

图 4-77　水平导航条

```
<div id="nav">
```

```
    <a href="#">首 页</a><a href="#">中心简介</a>
    <a href="#">政策法规</a><a href="#">常用下载</a>
    <a href="#">为您服务</a><a href="#">技术支持和服务</a>
</div>
```

CSS 样式代码如下：

```
<style type="text/css">
#nav{
    font-size: 14px;}
#nav a {
    color: red;
    background-color: #9CF;
    text-align: center;
    text-decoration: none;
    display: block;
    padding:6px 10px 4px;
    margin:0 2px;        /* 设置了左右边界,使两个 a元素间有 4px的水平间距 */
    border: 1px solid #39F;
    float:left;          /* 使 a元素浮动,实现水平排列 */  }
#nav a:hover {
    color: White;
    background-color: #930;  }
</style>
```

3. 制作栏目框标题栏

有时需要制作如图 4-78 所示的栏目框标题栏，标题栏的左端是栏目标题，右端是 more 之类的链接。如何将文字分别放置在一个盒子的左右两端呢？

|| 栏目标题1 more

图 4-78 栏目框标题栏

最简单的办法就是设置左边的文字左浮动、右边的文字右浮动。这时由于两个盒子都浮动，不占据外围容器的空间，所以必须设置外围盒子的高度，使它在视觉上能包含住两个浮动的盒子。结构代码如下：

```
<h3 id="colframe">
    <span class="title">栏目标题 1</span>
    <span class="more">more</span>
</h3>
```

CSS 代码如下：

```
#colframe {
    width:300px;
```

```
    margin:0 auto;
    border:1px gray solid;
    height:24px;        /* 由于浮动元素脱离了外围容器,必须使外围盒子高度伸展 */
    background-color:#CCCCCC;
    padding-top:10px;   /* 使文字垂直居中 */
}
#colframe span.title{
    float:left;
    padding-left:16px;   }
#colframe span.more{
    float:right;
    padding-right:12px;   }
```

另一种方法是让右边的元素不浮动,而是把它设置为块级元素,这样该元素的盒子就能伸展到整行,而浮动元素位于其左边,再设置它的内容右对齐,则效果一样。更改的代码如下:

```
#colframe span.more{
    display:block;
    text-align:right;
padding-right:12px;   }
```

第三种方法是将栏目标题写在 more 的后面,再设置 more 右浮动即可,要注意的是,这种方法必须将 more 写在前面,否则浮动元素会换行。

结构代码如下:

```
<h3 id="coltitle">
    <span class="more">>more</span>师生美文
</h3>
```

CSS 样式代码如下:

```
#coltitle span.more{
    float:right;
padding-right:12px;   }
```

栏目中的新闻标题和发布时间也是采用这种方式实现两端对齐的。

4. 1-3-1 固定宽度布局

在默认情况下,div 作为块级元素是占满整行从上到下依次排列的,但在网页的分栏布局中(例如 1-3-1 固定宽度布局),中间三栏(三个 div 盒子)必须从左到右并列排列,这时就需要将这三个 div 盒子都设置为浮动。

但三个 div 盒子都浮动后,只能浮动到窗口的左边或右边,无法在浏览器中居中。因此需要在三个 div 盒子外面再套一个盒子(称为 container),让 container 居中,这样就实现了三个 div 盒子在浏览器中居中,如图 4-79 所示。

header body
container
nav content side
pagefooter

图 4-79　1-3-1 布局示意图

注意：对于 Firefox 来说，由于 container 里面的三个盒子都浮动，脱离了标准流，所以都没有占据 container 容器的空间。从结构上看，应该是 container 位于三个盒子的上方，但这并不妨碍用 container 控制里面浮动的盒子居中。由于 container 占据的高度为 0，所以在 Firefox 等浏览器中的都看不到 container 的存在。而对于 IE 来说，由于 container 一般设置了宽度，作为网页的宽度，所以在 IE 中 container 会包含住三个盒子。

下面是 1-3-1 固定宽度布局的参考实现代码。效果如图 4-80 所示。

```
<style type="text/css">
#header,#pagefooter,#container{
    margin:0 auto;                  /* 与 width 配合实现水平居中 */
    width:772px;
    border: 1px dashed #FF0000;     /* 添加边框为演示需要 */
}
#navi,#content,#side{
    border:2px solid #0066FF;       /* 添加边框为演示需要 */
    float:left;                     /* 设置三栏都浮动 */
    width:200px;                    /* 设置三栏的宽度 */
}
#content{
    width:360px;                    /* 重新定义中间一栏的宽度 */
    }
#pagefooter{
    clear:both;                     /* 清除浮动,防止中间三列不等高时页脚顶上去 */
}
</style>
<body>
<div id="header">id="header"</div>
```

```
<div id="container">
    <div id="navi">id="navi"</div>
    <div id="content">id="content"</div>
    <div id="side">id="side"</div>
</div>
<div id="pagefooter">id="pagefooter"</div>
</body>
```



图 4-80　1-3-1 浮动方式布局效果图

制作 1-3-1 浮动布局的方法有很多种，实际上还可以将 pagefooter 块放到 container 块里面，这样设置 pagefooter 清除浮动后，在 IE 6 和 Firefox 中就都是 container 块包含住里面的三列和 pagefooter 块了。

提示：注意本例中浮动的三列 #navi、#content 和 #side 都没有设置 margin 属性，如果要为三列设置 margin 属性，以使三列之间有间隙，则最好给三列都增加一条 display:inline 的属性，否则 IE 6 浏览器会出现浮动盒子的双倍 margin 问题，导致最后一列容纳不下而移到下一行去。

4.7　相对定位和绝对定位

利用浮动属性定位只能使元素浮动形成图文混排或块级元素水平排列的效果，其定位功能仍不够灵活强大。本节介绍的在定位属性下的定位能使元素通过设置偏移量定位到页面或其包含框的任何一个地方，定位功能非常灵活。

4.7.1　定位属性和偏移属性

为了让元素在定位属性下定位，需要对元素设置定位属性 position，position 的取值有四种，即 relative、absolute、fixed 和 static。其中 static 是默认值，表示不使用定位属性定位，也就是盒子按照标准流或浮动方式排列。fixed 称为固定定位，它和绝对定位类似，只是总是以浏览器窗口为基准进行定位，但 IE 6 浏览器不支持该属性值。因此定位属性的取值中用得最多的是相对定位(relative)和绝对定位(absolute)，本节主要介绍它们的作用。

偏移属性是指 top、left、bottom、right 四个属性，为了使元素在定位属性下从基准位置发生偏移，偏移属性必须和定位属性配合使用，left 指相对于定位基准的左边向右偏移的值，top 指相对于定位基准的上边向下偏移的量。它们的取值可以是像素或百分比。如：

```
#mydiv {
    position:fixed;
    left: 50%;
    top: 30px;      }
```

注意：偏移属性仅对设置了定位属性的元素有效。

4.7.2 相对定位

使用相对定位的盒子的位置依据常以标准流的排版方式为基础，然后使盒子相对于它原来的标准位置偏移指定的距离。相对定位的盒子仍在标准流中，它后面的盒子仍以标准流方式对待它。

如果对一个元素定义相对定位属性(position: relative;)，那么它将保持在原来的位置上不动。如果再对它通过 top、left 等属性值设置垂直或水平偏移量，那么它将"相对于"它原来的位置发生移动。例如图 4-81 中的 em 元素就是通过设置相对定位再设置位移让它"相对于"原来的位置向左下角偏移，同时它原来的位置仍然不会被其他元素占据。代码如下：

```
em {
    background-color: #0099FF;
    position: relative;
    left: 60px;
    top: 30px;      }
p {
    padding: 25px;
    border: 2px solid #933; background-color: #dbfdba;}
<p>在远古时代,<em>人类与神都同样居住在地上</em>,一起过着和平快乐的日子,可是人类愈来愈聪明,不但学会了建房子、铺道路,还学会钩心斗角、欺骗等等不好的恶习,搞得许多神仙都受不了,纷纷离开人类,回到天上居住。</p>
```

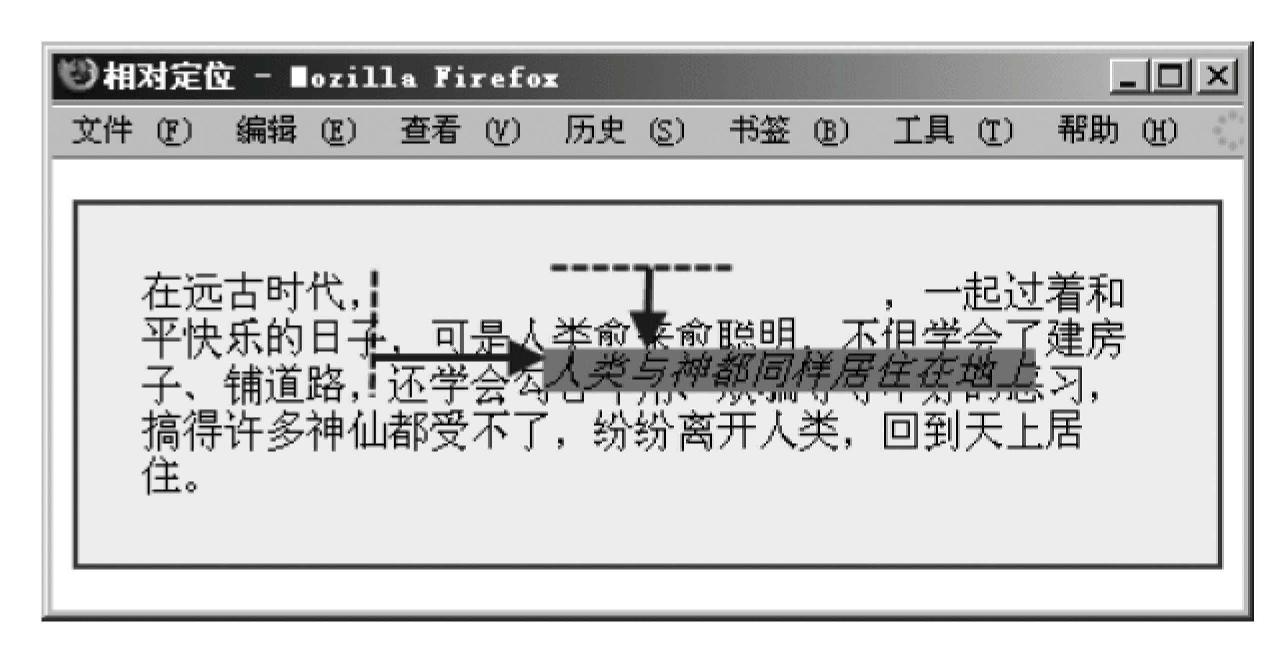


图 4-81　设置 em 元素为相对定位

可以看到元素设置为相对定位后有两点会发生：

(1) 元素原来占据的位置仍然会保留，也就是说，相对定位的元素未脱离标准流；

(2) 因为是使用了定位属性的元素，所以会和其他元素发生重叠。

设置元素为相对定位的作用可归纳为两种：一是让元素相对于它原来的位置发生位

移,同时不释放原来占据的位置;二是让元素的子元素以它为定位基准进行定位,同时它的位置保持不变,这时相对定位的元素成为包含框,一般是为了帮助里面的元素进行绝对定位。

4.7.3 相对定位的应用举例

1. 鼠标滑过时向右下偏移的链接

在有些网页中,当鼠标滑动到超链接上方时,超链接的位置会发生细微的移动,如向左下方偏移,让人觉得链接被鼠标拉上来了,如图 4-82 所示。

首 页 中心简介 政策法规 常用下载 为您服务

图 4-82 偏移的超链接(当鼠标悬停时向左下方偏移)

这种效果的制作原理其实很简单,主要就是运用了相对定位。在 CSS 中设置 a 元素为相对定位,当鼠标滑过时,就让它相对于原来的位置发生偏移。CSS 代码如下:

```
a:hover {
    color: red;
    position: relative;
    right: 2px;
    top: 3px; }
```

还可以给这些链接添加盒子,那么盒子也会按上述效果发生偏移,如图 4-83 所示。

首 页 中心简介 政策法规 常用下载 为您服务

图 4-83 给链接添加盒子,同样会偏移

2. 利用相对定位制作简单的阴影效果

在 4.5.3 节中,即使制作如图 4-48 所示的简单阴影效果都需要用到一张“左上边”的图片。我们可以利用相对定位技术,不用一张图片也能制作出和图 4-48 相同的简单阴影效果。它的原理是在 img 元素外套一个外围容器,将外围容器的背景设置为灰色,作为 img 元素的阴影,同时不设置填充边界等值使外围容器和图片一样大,这时图像就正好把外围容器的背景完全覆盖。再设置图像相对于原来的位置往左上方偏移几个像素,这样图像的右下方就露出了阴影盒子右边和下边部分的背景,看起来就是 img 元素的阴影了。代码如下,效果如图 4-48 所示。

```
.shadow img {
    padding: 6px;
    border: 1px solid #465B68;
    background-color: #fff;
    position: relative;
    left: -5px;
```

```
    top: -5px;      }
div.shadow {
    background-color: #CCC;
    float:left;      /* 使 div 盒子收缩,和 img 一样大 */      }
<div class="shadow"><img src="works.jpg" /></div>
```

3. 固定宽度网页居中的相对定位法

使用相对定位法可以实现固定宽度的网页居中,该方法首先将包含整个网页的包含框 container 进行相对定位使它向右偏移浏览器宽度的 50%,这时左边框位于浏览器中线的位置上,然后使用负边界将它向左拉回整个页面宽度的一半,如图 4-84 所示,从而达到水平居中的目的。代码如下:

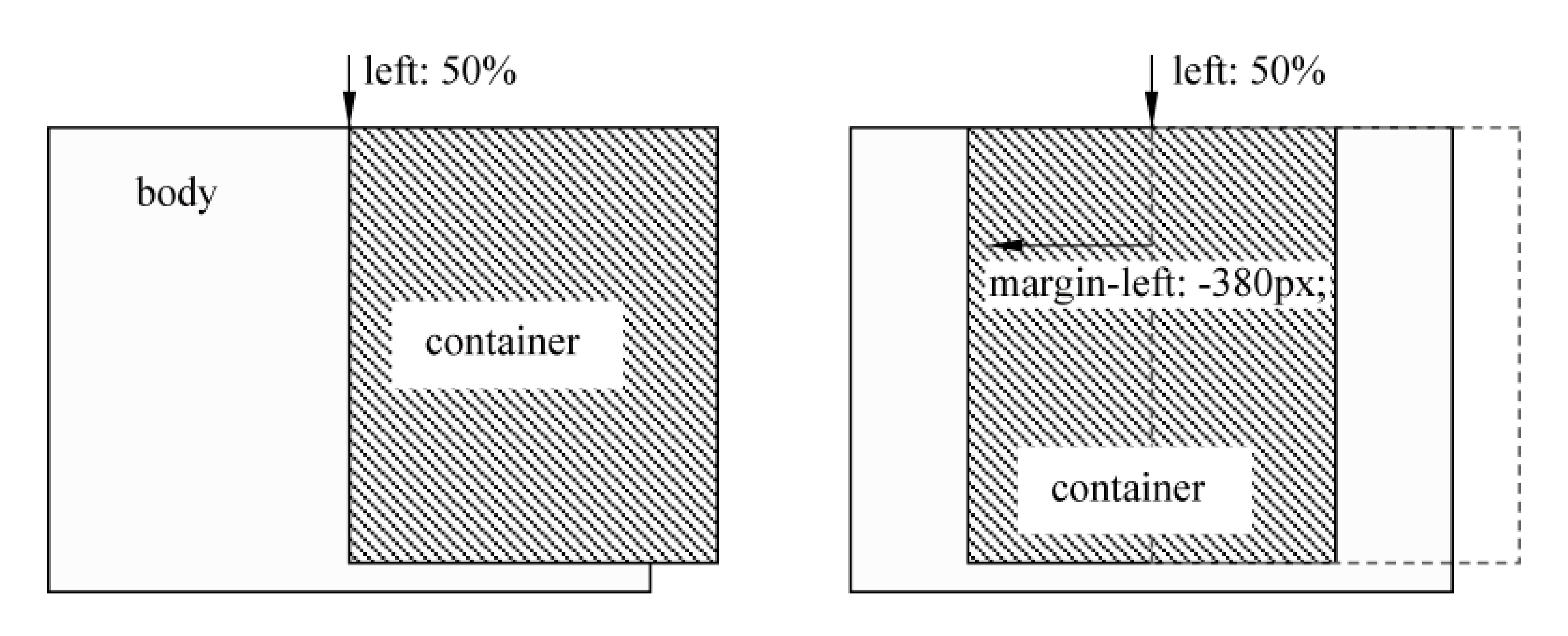


图 4-84 相对定位法实现网页居中示意图

```
#container {
    position:relative;
    width:760px;
    left:50%;
    margin-left:-380px;      }
```

这段代码的意思是,设置 container 的定位是相对于它原来的位置,而它原来默认的位置是在浏览器窗口的最左边,然后将其左边框移动到浏览器的正中央,这是通过"left: 50%"实现的,这样就找到了浏览器的中线。再使用负边界法将盒子的一半宽度从中线位置拉回到左边,从而实现了水平居中。

想一想: 如果把 # container 选择器中的"left: 50%; margin-left:-380px;"改为"right:50%; margin-right: -380px;",还能实现居中吗?

另外,大家知道 div 中的内容默认情况下是顶端对齐的,有时希望 div 中内容垂直居中,如果 div 中只有一行内容,可以设置 div 的高度 height 和行高 line-height 相等。而如果 div 中有多行内容,更一般的方法就是上面这种相对定位的思想,把 div 中的内容放入到一个子 div 中,让子 div 相对于父 div 向下偏移 50%,这样子 div 的顶部就位于父 div 的垂直中线上,然后再设置子 div 的 margin-top 为其高度一半的负值。

4.7.4 绝对定位

绝对定位是指盒子的位置以它的包含框为基准进行定位。绝对定位的元素完全脱离

标准流中。这意味着它们对其他元素的盒子定位没有影响，其他的元素就好像这个绝对定位元素完全不存在一样。

注意：绝对定位是以它的包含框的边框内侧为基准进行定位的，因此改变包含框的填充值不会对绝对定位元素的位置造成任何影响。

绝对定位的偏移值是指从它的包含框边框内侧到元素的外边界之间的距离，如果修改元素的 margin 值会影响元素内容的显示位置。

例如，如果将相对定位例子中的 em 的定位属性值由 relative 改为 absolute，那么 em 将按照绝对定位方式进行定位，从图 4-85 中可以看出它将以浏览器左上角为基准定位，配合 left、top 属性值进行偏移，同时 em 元素原来所占据的位置将消失，也就是说，它脱离了标准流，其他元素当它不存在了一样。em 选择器的代码如下：

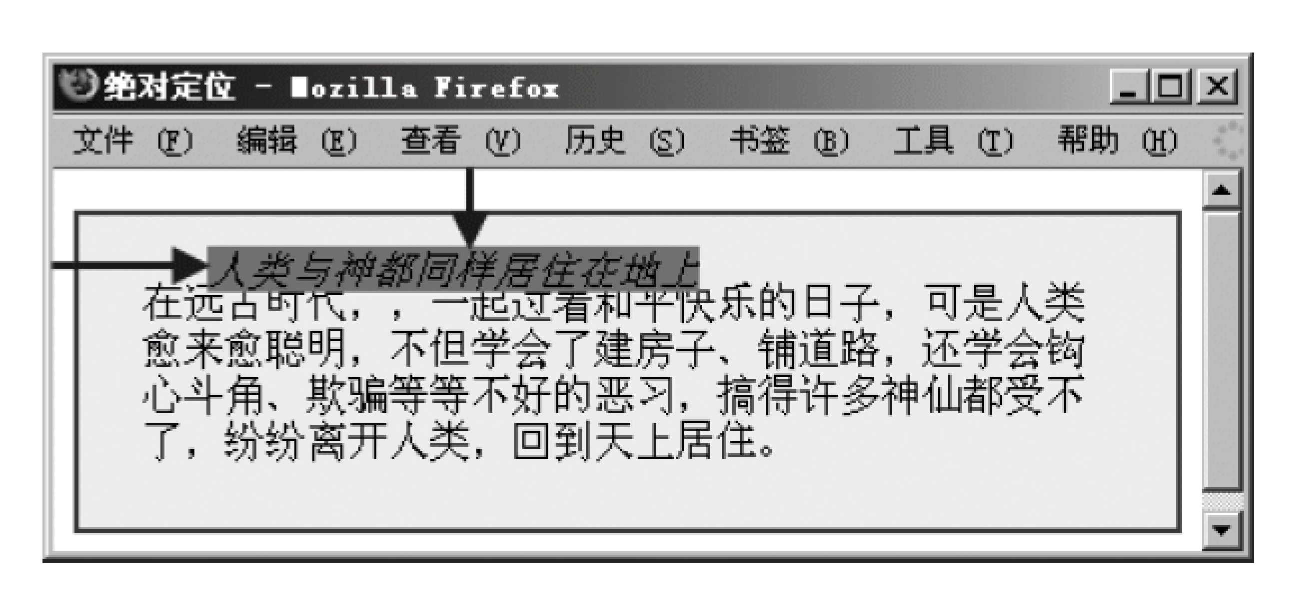


图 4-85　设置 em 元素为绝对定位

```
em {
    background-color: #0099FF;
    position:absolute;
    left: 60px;
    top: 30px;       }
```

但要注意的是，设置为绝对定位(position:absolute;)的元素，并非总是以浏览器窗口为基准进行定位的。实际上，绝对定位元素是以它的包含框为基准进行定位的，所谓包含框，是指距离它最近的设置了定位属性的父级元素的盒子。如果它所有的父级元素都没有设置定位属性，那么包含框就是浏览器窗口。

下面对 em 元素的父级元素 p 设置定位属性，使 p 元素成为 em 元素的包含框。这时，em 元素就不再以浏览器窗口为基准进行定位了，而是以它的包含框 p 元素的盒子为基准进行定位，效果如图 4-86 所示。

对应的 CSS 代码如下：

```
p {
    background-color: #dbfdba;
    padding: 25px;
    position:relative;                /* 让 p元素成为包含框 */
    border: 2px solid #6c4788;       }
em {
    background-color: #0099FF;
    position:absolute;
```

```
left: 60px;
top: 40px;      }
```

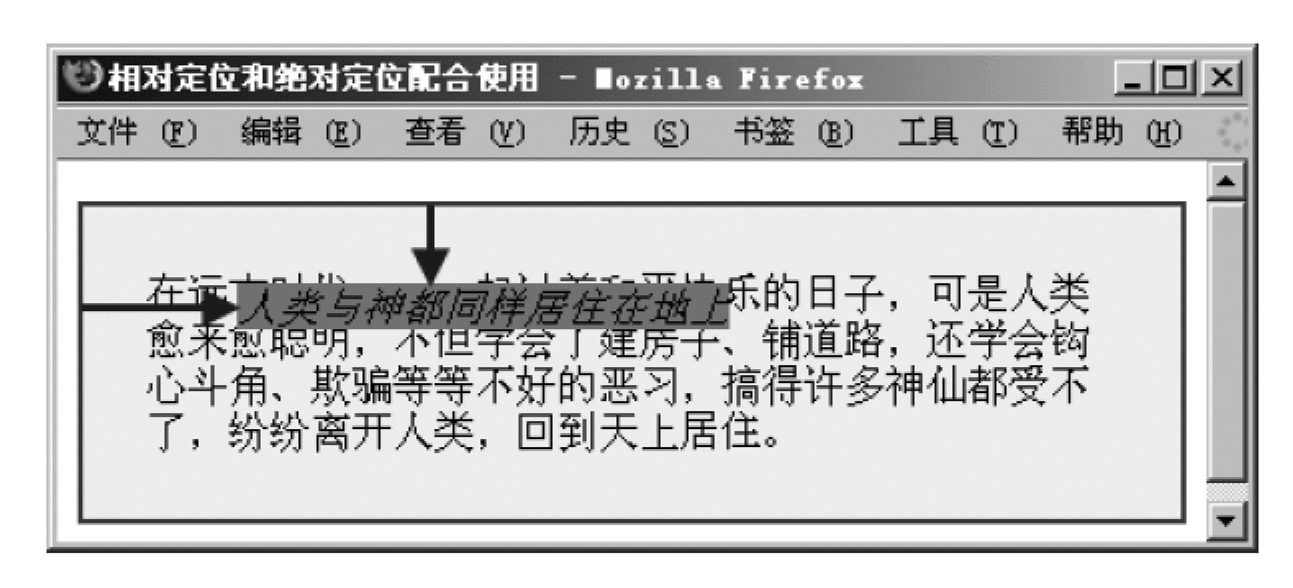


图 4-86　设置 em 为绝对定位同时设置 p 为相对定位

上述代码就是相对定位和绝对定位配合使用的例子，这种方式非常有用，可以让子元素以父元素为定位基准进行定位。

表 4-6 对相对定位和绝对定位的特点进行了比较。

表 4-6　相对定位和绝对定位的比较

	相对定位(relative)	绝对定位(absolute)
定位基准	以该元素原来的位置为基准	元素以距离它最近的设置了定位属性的父级元素为定位基准，若它所有的父级元素都没设置定位属性，则以浏览器窗口为定位基准
原来的位置	还占用着原来的位置，未脱离标准流	不占用其原来的位置，已经脱离标准流，其他元素就当它不存在一样
宽度	盒子的宽度不会收缩	盒子的宽度会自动收缩

4.7.5　绝对定位的应用举例

绝对定位元素的特点是完全脱离了标准流，不占据网页中的位置，而是浮在网页上。利用这个特点，绝对定位可以制作漂浮广告、弹出菜单等浮在网页上的元素。如果希望绝对定位元素以它的父元素为定位基准，则需要对它的父元素设置定位属性(一般都是设置为相对定位)，使它的父元素成为包含框，这就是绝对定位和相对定位的配合使用。这样就可以制作出缺角的导航条、小提示窗口或下拉菜单等。

1. 制作缺角的导航条

图 4-87 是一个缺角的导航条，这是一个利用定位基准和绝对定位技术结合的典型例子，下面来分析它是如何制作的。

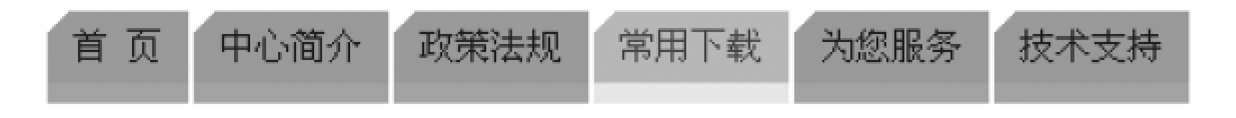


图 4-87　缺角的导航条

首先，如果这个导航条没有缺角，那么这个水平导航条完全可以通过盒子在标准流及

浮动方式下的排列来实现，不需要使用定位属性。其次，缺的这个角是通过一个元素的盒子叠放在导航选项盒子上实现的，它们之间的位置关系如图 4-88 所示。

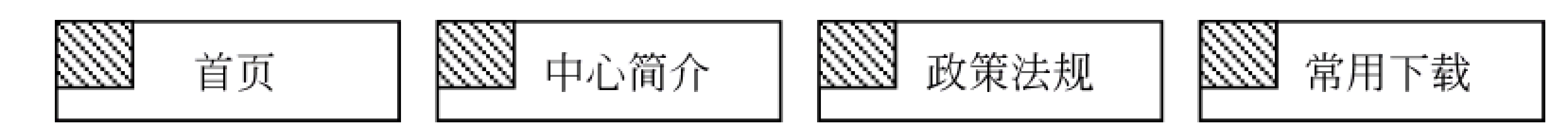


图 4-88 缺角的导航条元素盒子之间的关系

形成缺角的盒子实际上是一个空元素，该元素的左边框是 8 像素宽的白色边框，下边框是 8 像素宽的蓝色边框，它们交会形成了斜边效果，如图 4-89 所示。



图 4-89 缺角处是一个左白、下蓝边框的空元素

可以看出，导航项左上角的盒子必须以导航项为基准进行定位，因此必须设置导航项的盒子为相对定位，让它成为一个包含框，然后将左上角的盒子设置为绝对定位，使左上角的盒子以它为基准进行定位，这样还能使左上角盒子不占据标准流的空间。同时由于导航条不需要改变在标准流中的位置，所以应该设置为相对定位无偏移。

下面将这个实例分解成几步来做：

(1) 首先写出结构代码，我们直接用 a 元素的盒子做导航条，因为 a 元素里面还要包含了一个盒子，所以应在 a 元素中添加任意一个行内元素，这里选择 b 元素，它的内容应为空，这样才能利用盒子的边框交会做三角形。结构代码如下：

```
<div id="nav4">
    <a href="#"><b></b>首 页</a>
      <a href="#"><b></b>中心简介</a>
      …
      <a href="#"><b></b>技术支持</a>
</div>
```

(2) 因为要设置 a 元素的边框填充等值，所以设置 a 元素为块级元素显示，而要让块级元素水平排列，必须设置这些元素为浮动。当然，设置为浮动后元素将自动以块级元素显示，因此也可以将 a 元素的“display:block;”去掉。同时，要让 a 元素成为其子元素的包含框，必须设置 a 元素的定位属性，而 a 元素应保持它在标准流中的位置不发生移动，所以 a 元素的定位属性值应为 relative。因此，a 元素的 CSS 代码如下：

```
#nav4 a {
    background-color: #79bcff;
    font-size: 14px;   color: #333;
    text-decoration: none;
    border-bottom:8px solid #99CC00;     /* 以上 5条为普通 CSS样式设置 */
    display: block;
    float: left;
    padding: 6px 10px 4px 10px;
    margin:0 2px;
```

```
    position:relative;                /* 让 a元素作为 b元素的定位基准 */  }
```

(3) 接下来设置 b 元素为绝对定位,让它以 a 元素为包含框进行定位。由于 b 位于 a 的左上角,必须设置偏移属性"left:0;"和"top:0;"。由于 b 元素还没有内容,所以此时看不见 b 元素。再设置 b 元素的左边框为白色,下边框为 a 元素的背景色。这样在 Firefox 中就可以看见缺角的导航条效果了。为了在 IE 中也有此效果,需要设置"overflow: hidden;"和"height: 0px;",因为 IE 在默认情况下,设置了边框属性的空元素也有 12px 的高度,所以 b 元素的 CSS 代码如下:

```
#nav4 a b {
    border-bottom: 8px solid #79bcff;
    border-left: 8px solid #ffffff;   /* 左边框和下边框交会形成三角形效果 */
    overflow: hidden;
    height: 0px;                      /* 以上两条为了兼容 IE 6,使空元素高度为 0 */
    position: absolute;
    left:0;                           /* 相对于 a元素边框内侧的左上角定位 */
    top:0;      }
```

(4) 最后为导航条添加交互效果,只需设置鼠标经过时 a 元素的字体、背景色改变,b 元素下边框颜色改变就可以了。

```
#nav4 a:hover {
    color: #c00;
    background-color: #ccc;
    border-bottom-color: #cf3;      }
#nav4 a:hover b {
    border-bottom-color: #ccc;      }
```

这样,这个缺角的导航条就制作完成了。网上还有很多这种带有三角形的导航条,例如在图 4-90 中,只是在默认状态时将三角形隐藏,而鼠标滑过时显示三角形罢了。

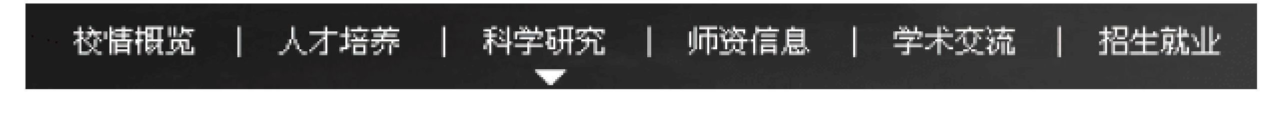


图 4-90　带有三角形的导航条

2. 制作小提示窗口(4-6.html)

我们知道,几乎所有的 HTML 标记都有一个 title 属性。添加该属性后,当鼠标停留在元素上时,会显示 title 属性里设置的文字。但用 title 属性设置的提示框不太美观。实际上,可以用绝对定位元素来模拟小提示框,由于这个小提示框必须在其解释的文字旁边出现,所以要把待解释的文字设置为相对定位,作为小提示框的定位基准。

下面是 CSS 小提示框的代码,它的显示效果如图 4-91 所示。

```
<style type="text/css">
a.tip{
    color:red;
```

```
        text-decoration:none;
        position:relative;          /* 设置待解释的文字为定位基准 */
    }
    a.tip span {display:none;}      /* 默认状态下隐藏小提示窗口 */
    a.tip:hover {cursor:hand;       /* 当鼠标滑过时将鼠标指针设置为手形 */
    z-index:999;}
    a.tip:hover .popbox {
        display:block;              /* 当鼠标滑过时显示小提示窗口 */
        position:absolute;
        top:15px;
        left:-30px;
        width:100px;                /* 以上三条设置小提示窗口的显示位置及大小 */
        background-color:#424242;
        color:#fff;
        padding:10px;
    z-index:9999;                   /* 设置很大的层叠值防止被其他 a 元素覆盖 */
    }
    p {  font-size: 14px;  }
    </style>
    <body><p>Web 前台技术:<a href="#" class="tip">Ajax<span class="popbox">Ajax 是一种浏览器无
    刷新就能和 Web 服务器交换数据的技术</span></a>技术和
    <a href="#" class="tip">CSS<span class="popbox">Cascading Style Sheets 层叠样式表</span></a>
   的关系</p></body>
```

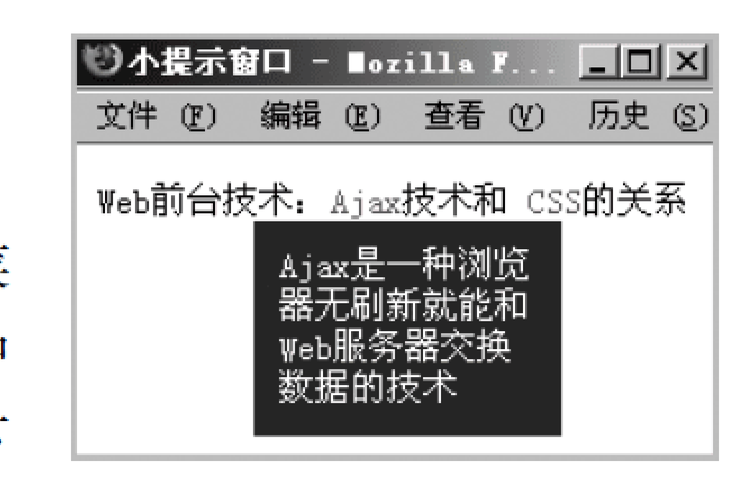


图 4-91 小提示窗口的效果

3. 制作纯 CSS 下拉菜单

下拉菜单是网页中常见的高级界面元素，过去下拉菜单一般都用 JavaScript 制作。例如，使用 Dreamweaver 中的“行为”或在 Fireworks 中“添加弹出菜单”都可以制作下拉菜单，它们是通过自动插入 JavaScript 代码实现的，但这些软件制作的下拉菜单存在代码复杂、界面不美观等缺点，因此现在更推荐使用 CSS 来制作下拉菜单，它具有代码简洁、界面美观、占用资源少的特点。

下拉菜单的特点是弹出时浮在网页上的，不占据网页空间，所以放置下拉菜单的元素必须设置为绝对定位元素，而且下拉菜单的位置是依据它的导航项来定位的，所以导航项应该设置为相对定位，作为下拉菜单的定位基准。在默认状态下，设置下拉菜单元素的 display 属性为 none，使下拉菜单被隐藏起来。当鼠标滑到导航项时，显示下拉菜单。

制作下拉菜单的步骤比较复杂，下面一步步来完成：

(1) 下拉菜单采用二级列表结构，第一级放导航项，第二级放下拉菜单项。首先写出它的结构代码，此时显示效果如图 4-92 所示。

```
    <ul id="nav">
```

```
<li><a href="">文 章</a>
  <ul>
    <li><a href="">Ajax教程</a></li>
                    …
    <li><a href="">Flex教程</a></li>
  </ul>
  </li>
<li><a href="">参 考</a>
  <ul>
    <li><a href="">E-cash</a></li>
    <li><a href="">微支付</a></li>
    <li><a href="">混沌加密</a></li>
  </ul>
</li>
<li><a href="">Blog</a>
  <ul>
    <li><a href="">生活随想</a></li>
                    …
    <li><a href="">随意写</a></li>
  </ul>
</li>
</ul>
```

可以看到，下拉菜单被写在内层的 ul 里，只需控制这个 ul 元素的显示和隐藏就能实现下拉菜单效果。

(2) 设置第一层 li 为左浮动，这样导航项就会水平排列，同时去除列表的小黑点、填充和边界。此时显示效果如图 4-93 所示。再设置导航项 li 为相对定位，让下拉菜单以它为基准定位。代码如下：

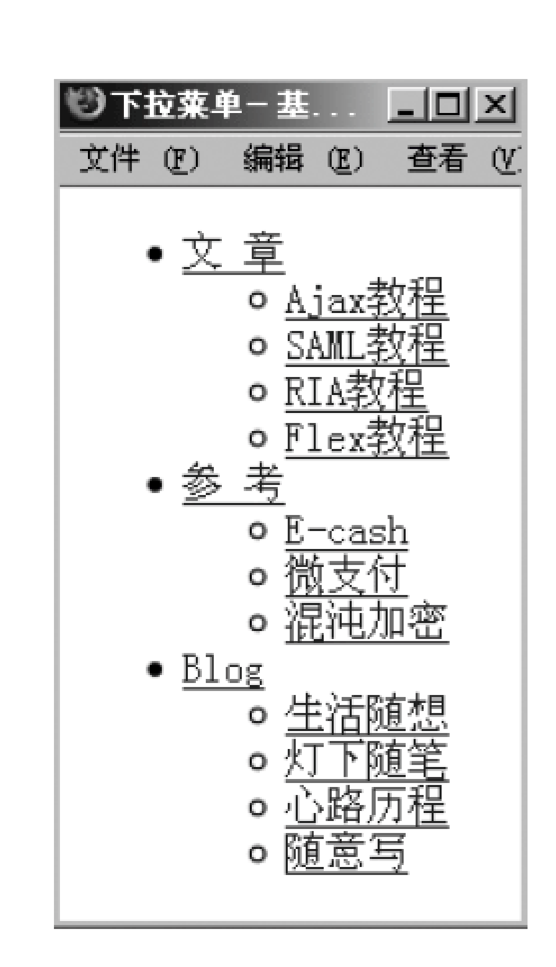


图 4-92　下拉菜单基本结构

```
#nav, #nav ul {
    padding: 0;   margin: 0;
    list-style: none;     }
li {
    float: left;
    width: 160px;
    position:relative;     }
```



图 4-93　下拉菜单水平排列—设置第一级 li 左浮动

（3）设置下拉菜单为绝对定位，位于导航项下 21px。默认状态下隐藏下拉菜单 ul，所以 ul 默认值是不显示的。

```
li ul {
    display: none;
    position: absolute;
    top: 21px;     }
```

再添加交互，当鼠标滑过时显示下拉菜单 ul。此时在 Firefox 中就可以看到鼠标滑过时弹出下拉菜单的效果了，如图 4-94 所示，只是不太美观。

```
li:hover ul {          /* IE 6不支持非 a元素的伪类,故 IE 6不显示下拉菜单 * /
    display: block;     }
```



图 4-94　添加了交互的下拉菜单—当鼠标滑过时显示下拉菜单项

（4）最后改变下拉菜单的 CSS 样式，使它更美观，并添加交互效果，代码如下。最终在 Firefox 中的效果如图 4-95 所示。

```
ul li a{
    display:block;
    font-size:14px; color: #333;         /* 设置文字效果 * /
    text-align:center;
    text-decoration: none;
    border: 1px solid #ccc;
    padding:3px;
    height:1em;                          /* 解决 IE 6的 bug* /
}
ul li a:hover{
    background-color:#f4f4f4;
    color:red;     }
```



图 4-95　对下拉菜单进行美化后的效果

想一想：如果把上述选择器中的(position:relative;)和(position:absolute;)都去掉还会有上面的下拉菜单效果吗？会出现什么问题呢？

(5) 使下拉菜单兼容IE 6浏览器的基本思想。

由于IE 6浏览器不支持li:hover伪类，所以无法弹出菜单。一种兼容IE 6浏览器的方法是在网页head部分插入下面一段JavaScript代码。代码如下：

```
<script>
startList=function() {
        navRoot=document.getElementById("nav");
        node=navRoot.getElementsByTagName("li");
    for(i=0; i<node.length; i++) {
        node[i].onmouseover=function() {
            this.className+=" over";        //over前面有个空格不能省略
                }
        node[i].onmouseout=function() {
            this.className=this.className.replace(/over/,"");
                }}}
window.onload=startList;
</script>
```

添加一个CSS选择器，代码如下。使JavaScript能动态地为li元素添加、移除“.over”这个类，从而控制“li ul”的显示和隐藏。

```
li.over ul {  display: block;  }
```

4. hover伪类的应用总结

hover伪类是通过CSS实现与页面交互的最主要形式，本节的所有实例中都用到了hover伪类，下面总结一下hover伪类的作用。

hover伪类的作用有两种：一是定义元素在鼠标滑过时样式的改变，以实现动态效果，这是hover伪类的基本用法，如鼠标滑过导航项时让导航项的字体和背景变色等。

二是通过hover伪类控制子元素的动态效果。用hover伪类控制元素的子元素又可分为两种情况：

(1) 解决IE 6不支持非a元素hover伪类的问题。

由于IE 6只支持a:hover伪类，如果要给其他元素添加动态效果，就可以在该元素外面套一个a元素，例如在img元素外套一个a元素，就可以用a:hover img来设置鼠标滑过img时的动态效果了。

(2) 控制子元素的显示和隐藏。

有时如果子元素通过display:none隐藏起来了，就没有办法利用子元素自身的hover伪类来控制它了，只能使用父元素的hover伪类对它进行控制，例如下拉菜单。

hover伪类不能做什么：hover伪类只能控制元素自身或其子元素在鼠标滑过时的动态效果，而无法控制其他元素实现动态效果，例如tab面板由于要用tab项(a元素)控制不属于其包含的div元素，就无法使用hover伪类实现，而只能通过编写JavaScript代

码来操纵 a 元素的行为实现。

4.7.6 与定位属性有关的 CSS 属性

CSS 中，有几个属性只有在元素设置了定位属性(position)之后才有效，例如，z-index 属性、偏移属性(top、left 等)和裁切属性等。

在 Dreamweaver 中，对这些与 position 相关的属性设置在“定位”选项面板中，其中，“宽”和“高”对应 width 和 height 属性，实际上这两项的设置在“方框”面板中也有。“裁切”可用来对图像或其他盒子进行剪切，但仅对绝对定位元素有效。“显示”对应 visibility 属性，若设置为隐藏，则元素不可见，但元素所占的位置仍然会保留。

1. z-index 属性

z-index 属性用于调整定位时重叠块之间的上下位置。与它的名称一样，想象页面为 x-y 轴，那么垂直于页面的方向就为 z 轴，z-index 值大的盒子会叠放在值小的盒子的上方，可以通过设置 z-index 值改变盒子之间的重叠次序。z-index 默认值为 0，当两个盒子的 z-index 值一样时，则保持原来的高低覆盖关系。

2. 制作动态改变叠放次序的导航条

利用 z-index 属性改变盒子叠放次序的功能，我们可以制作出如图 4-96 所示的导航条来。该导航条由若干个导航项和下部的水平条组成。水平条是一个绝对定位元素，通过设置它的位置使它正好叠放在导航项下面的部分上。在正常浏览状态下，导航项的下方被水平条覆盖，当鼠标滑过某个导航项时，设置它的 z-index 值变大，这样该导航项就会遮盖住水平条，形成如图 4-96 所示的动态效果。



图 4-96 动态改变 z-index 属性的导航条

下面分步来讲解如何制作动态改变 z-index 属性的导航条。

(1) 首先，因为 z-index 只对设置了定位属性的元素才有效，所以导航项和水平条都要设置定位属性。由于每个导航项的位置应该保持在标准流中的位置不变，所以设置它们为相对定位，不设置偏移属性。而水平条要叠放在导航项的上方，不占据网页空间，因此设置它为绝对定位。而且水平条要以整个导航条为基准进行定位，所以将整个导航条放在一个 div 盒子内，并设置它为相对定位，作为水平条的定位基准。结构代码如下：

```
<div id="nav">                          <!--主要作用是作为底部水平条的定位基准 -->
    <a href="#"><span>首 页</span></a>              <!--该导航条使用了滑动门技术 -->
    <a href="#"><span>中心简介</span></a>   …
    <a href="#"><span>技术支持</span></a>
    <div id="bott"></div>               <!--底部的水平条 -->
</div>
```

(2) 接下来写导航条#nav 和它包含的水平条的 CSS 代码。#nav 只要设置为相对

定位就可以了，作为水平条＃bott 的定位基准，而＃bott 设置为绝对定位后必须向下偏移 28px，这样正好叠放于导航项的下部。

```
#nav {  position:relative;        /* 作为定位基准 */  }
#bott{
    background-color: #999966;
    height:6px;                   /* 水平条高度为 6px */
    font-size:0;                  /* 兼容 IE,也可用 overflow:hidden 替代 */
    clear:both;                   /* 由于 a 元素都浮动,所以要清除浮动 */
    position:absolute;
    width:95%;                     /* 绝对定位元素宽度不会自动伸展,设置宽度使其占满一行 */
    top:28px;      }
```

(3) 用滑动门技术设置 a 元素和 span 元素的背景，背景图片如图 4-97 所示。其中 span 元素的背景从右往左铺，a 元素的背景从左往右铺，叠加后形成自适应宽度的圆角导航项背景。再设置 a 元素为相对定位，这是为了使 a 元素在鼠标滑过时能设置 z-index 属性。代码如下：

```
#nav a {
    position:relative;            /* 设置为相对定位,为了应用 z-index 属性 */
    float: left;                  /* 使 a 元素水平排列 */
    padding-left: 14px;
    background: url(images/zindex.gif) 0 -42px;              /* 取下半部分的图案作背景 */
    height:34px;
    line-height:28px;             /* 行高比高度小,使文字位于中部偏上 */
    color:White;
    text-decoration:none;   }
#nav span {
    padding-right:14px;
    background: url(images/zindex.gif) 100%-42px;
    font-size:14px;
    float:left;                   /* 此处是为兼容 IE 6,防止 span 占满整行 */   }
```

图 4-97 导航条的背景图片(zindex. gif)

(4) 最后设置鼠标滑过时的效果，包括设置 z-index 值改变重叠次序，改变背景显示位置实现图像的翻转等。代码如下：

```
#nav a:hover {
    cursor:hand;                  /* 使 IE 6中光标变为手形 */
    background-position:0 0;      /* 取上半部分图像作为背景 */
```

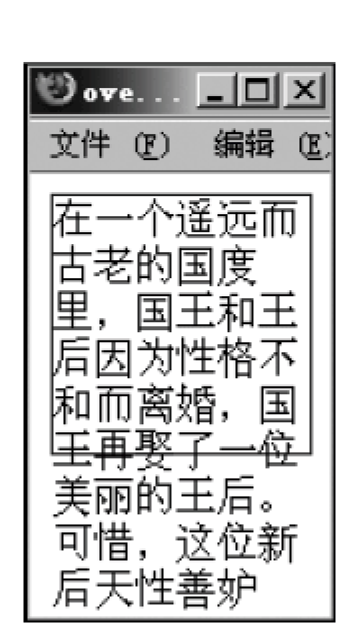
```
    z-index:1000;                /* 使鼠标悬停的导航项遮盖住水平条 */}
#nav a:hover span {
    height:34px;
    background-position:100% 0;  /* 取下半部分图像作为背景,实现背景的翻转 */
    color:#ff0000;     }
```

这样动态改变层叠次序的导航条就做好了,如果将导航条的背景图片制作成具有半透明效果的 png 格式文件,效果可能会更好。

3. overflow 属性

(1) overflow 属性的基本功能是设置元素盒子中的内容如果溢出是否显示,取值有 visible(可见)、hidden(隐藏)、scroll(出现滚动条)、auto(自动)。如果不设置则默认值为 visible。将下面代码中的 overflow 值依次修改为 visible、hidden、scroll、auto 的显示效果如图 4-98 所示。

```
<style type="text/css">
#qq {
    border:1px solid #333333;
    height: 100px;
    width: 100px;
    overflow: visible;        /* 依次修改为 hidden、scroll、auto */}
</style>
<div id="qq">在一个遥远而古老的国度里,国王和王后因为性格不合而离婚,国王再娶了一位美丽的王后。可惜,这位新后天性善妒</div>
```



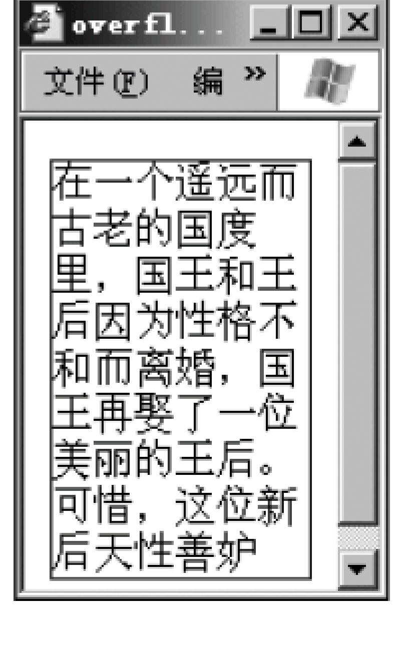








图 4-98　从左至右 overflow 属性分别设置为 visible(Firefox)、visible(IE 6)、hidden、scroll、auto 的效果

由于 IE 对于空元素的默认高度是 12px,所以经常使用(overflow:hidden)使空元素在 IE 浏览器中所占高度为 0。

(2) overflow 属性的另一种功能是用来代替清除浮动的元素。

如果父元素中的子元素都设置成了浮动,那么子元素脱离了标准流,导致父元素高度不会自动伸展包含住子元素,在前面介绍过可以在这些浮动的子元素的后面添加一个清除浮动的元素,来把外围盒子撑开。实际上,通过对父元素设置 overflow 属性也可以扩展外围盒子高度,从而代替了清除浮动元素的作用。例如:

```
<style type="text/css">
div{
    padding:10px;   margin:10px;
    border:1px dashed #111111;
    background-color:#90baff;   }
.father{
    background-color:#ffff99;
    border:1px solid #111111;
    overflow:auto;        /* 图 4-98(左)是未添加这句时的效果 */}
.son1{
    float:left;     }
</style>
<div class="father">
        <div class="son1">Box-1</div>
    </div>
```

可看到，对父元素设置 overflow 属性为 auto 或 hidden 时，就能达到在 Firefox 中扩展外围盒子高度的效果，如图 4-99(右)所示，这比专门在浮动元素后添加一个清除浮动的空元素要简单得多。





图 4-99 利用 overflow 属性扩展外围盒子高度之前(左)和之后(右)的效果

对于 IE 来说，只要设置浮动元素的父元素的宽或高，那么浮动元素就不会脱离标准流。父元素会自动伸展包含住浮动块，因此不存在扩展外围盒子高度的问题。

但当没有对父元素 box 设置宽或高时，在 IE 中父元素也不会包含住浮动块，而且对 IE 即使按上述方法设置父元素的 overflow 属性也不起作用。这时对 IE 来说，只能对盒子设置宽或高，如果不方便设置宽度，则可以针对 box 设置一个很小的百分比高度，如(height:1%)，使 IE 6 中的 box 也能包含住浮动块，这样就兼容了 IE 6 和 Firefox 浏览器。

另外，对浮动元素后面的元素设置 overflow:hidden 也能使 Firefox 出现和 IE 中相同的效果，读者可以对图 4-99 中对应的代码做修改来验证这一点。

4.8 CSS+div 布局

使用 CSS 布局时，先不要考虑网页的外观，而应该先思考网页内容的语义和结构。因为一个结构良好的 HTML 页面可以通过任何外观表现出来。

虽然普通用户看到的网页上有文字、图像等各种内容。但对于浏览器来说，它“看到”的页面内容就是大大小小的盒子。对于 CSS 布局而言，本质就是大大小小的盒子在页面上的摆放。我们看到的页面中的内容不是文字，也不是图像，而是一堆盒子。要考虑的就是盒子与盒子之间的关系，是上下排列、左右排列还是嵌套排列，是通过标准流定位还是通过浮动、绝对定位、相对定位实现，定位基准是什么等。将盒子之间通过各种定位方式排列使之达到想要的效果就是 CSS 布局的基本思想。

CSS 网页布局的基本步骤如下：

(1) 将页面用 div 分块；

(2) 通过 CSS 设计各块的位置和大小，以及相互关系；

(3) 在网页的各大 div 块中插入作为各个栏目框的小块。

表 4-7 对表格布局和 CSS＋DIV 布局的特点进行了比较。

表 4-7　表格布局和 CSS＋DIV 布局的比较

	表 格 布 局	CSS＋DIV 布局
布局方式	将页面用表格和单元格分区	将页面用 div 等元素分块
控制元素占据的页面大小	通过＜td＞标记的 width 和 height 属性确定	通过 CSS 属性 width 和 height 确定
控制元素在页面中的位置	在单元格前插入指定宽度的单元格使元素位置向右移动，或插入行或占位表格使元素向下移动	设置元素的 margin 属性或设置其父元素的 padding 属性使元素移动到指定位置
图片的位置	只能通过图片所在单元格的位置控制图片的位置	既可以通过图片所在元素的位置确定，又可以使用背景的定位属性确定图片的位置

4.8.1　分栏布局的种类

网页的布局从总体上说可分为固定宽度布局和可变宽度布局两类。所谓固定宽度，是指网页的宽度是固定的，如 980px，不会随浏览器大小的改变而改变；而可变宽度是指如果浏览器窗口大小发生变化，网页的宽度也会变化，例如将网页宽度设置为 85%，表示它的宽度永远是浏览器宽度的 85%。

固定宽度的好处是网页不会随浏览器大小的改变而发生变形，窗口变小只是网页的一部分被遮盖住，所以固定宽度布局用得更广泛，适合初学者使用。而可变宽度布局的好处是能适应各种显示器屏幕，不会因为用户的显示器过宽而使两边出现很宽的空白区域。以 1-3-1 式三列布局为例，它具有的布局形式如图 4-100 所示。

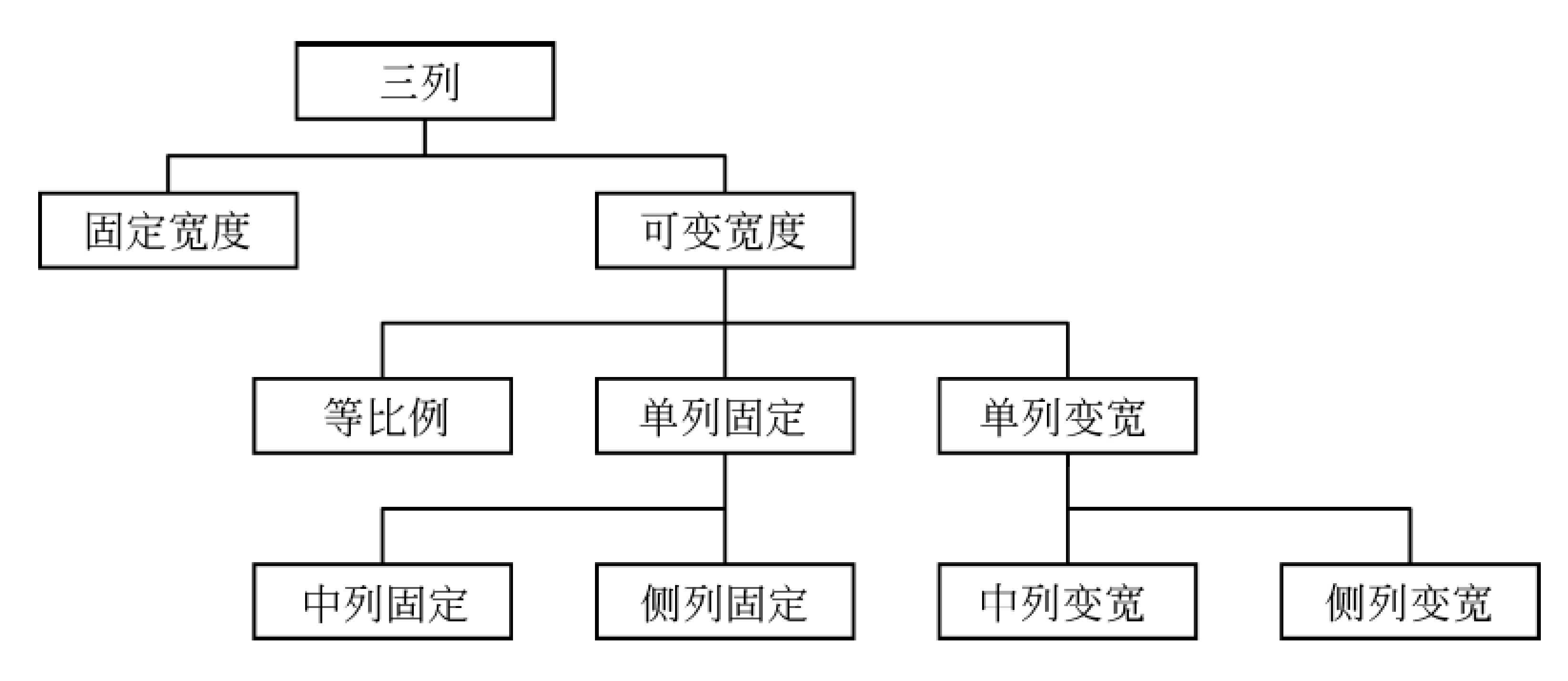


图 4-100　1-3-1 式布局所有的种类

4.8.2　固定宽度布局

1. 固定宽度分栏布局的实现

固定宽度布局的最常用方法是将所有栏都浮动，在 4.6.4 节关于已经介绍了三栏浮动实现 1-3-1 布局的方法，此处不再赘述。

2. 固定宽度网页居中的方法

通常情况下我们都希望制作的网页在浏览器中居中显示，通过 CSS 实现网页居中主要有以下三种方法：

1）text-align 法

这种方法设置 body 元素的 text-align 值为 center，这样 body 中的内容（整个网页）就会居中显示。由于 text-align 属性具有继承性，网页中各个元素的内容也会居中显示，这是我们不希望看到的，因此设置包含整个网页的容器 #container 的 text-align 值为 left。代码如下：

```
body{text-align: center;min-width: 990px;}
#container {margin: 0 auto;text-align: left;width: 990px;}
```

2）margin 法

通过设置包含整个网页的容器 #container 的 margin 值为“0　auto”，即上下边界为 0，左右边界自动，再配合设置 width 属性为一个固定值或相对值，也可以使网页居中，从代码量上看，这是使网页居中一种最简洁的办法。例如：

```
#container { margin: 0 auto;   width: 980px; }
#container { margin: 0 auto;   width: 85%; }
```

注意：如果仅设置 #container { margin：0　auto；}，而不设置 width 值，网页是不会居中的，而且使用该方法网页顶部一定要有文档类型声明 DOCTYPE，否则在 IE 6 中不会居中。

3）相对定位法

相对定位法居中在 4.7.3 相对定位的应用一节中已经介绍过，它只能使固定宽度的

网页居中。代码如下：

```
#container { position: relative;  width:980px;  left: 50%;  margin-left: -490px; }
```

4.8.3 CSS 布局的案例——重构太阳能网站

本节使用 CSS 布局的方法重新制作 2.6.7 节中用表格布局制作的太阳能网站，这称为网站重构。CSS 布局本质上就是设计盒子在页面上如何排列，图 4-101 是该网站 CSS 布局示意图，最终效果和图 2-45 中表格布局的网页效果完全相同。制作步骤如下。

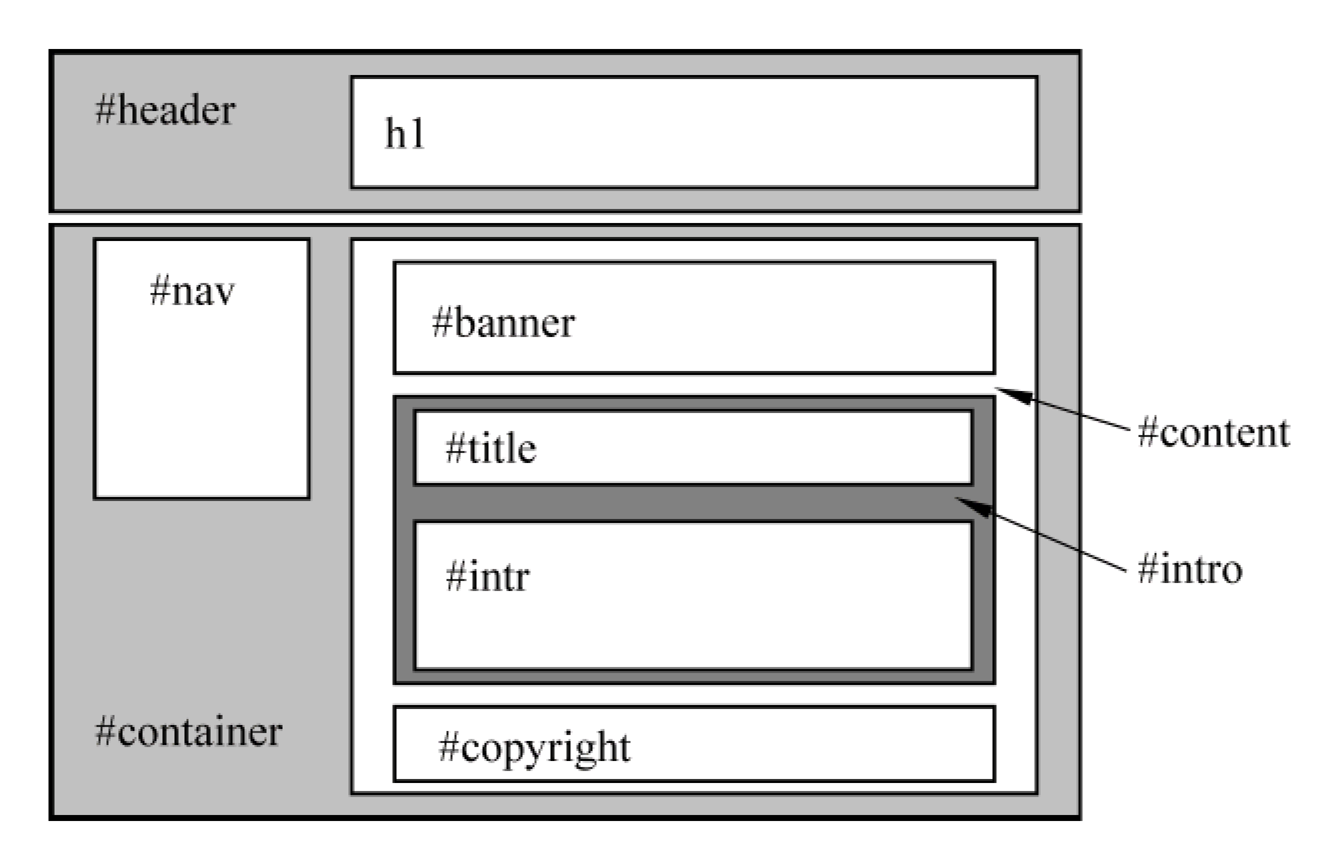


图 4-101　太阳能网站 CSS 布局示意图

1. 制作网页的头部

(1) 将网页划分为两部分，即上方的 header 部分和主体的 container 部分，如图 4-101 所示，观察 header 部分有两个背景色(绿色和白色)和一个背景图像，而一个元素的盒子最多只能设置一种背景色和一个背景图像，因此需要插入两个盒子来实现。代码如下：

```
<div id="header"><h1>光普太阳能网站</h1></div>
```

(2) 设置 #header 的背景色为绿色，宽为网页的宽度 852px。

```
#header{
    background-color:#99cc00;
    width:852px;    }
```

(3) 设置 h1 的背景色为白色，并设置背景图像为 logo.jpg，通过设置 margin 使盒子向右偏移 161px，然后用 text-indent 方法隐藏标记中的文字。这样网页的头部就做好了。

```
#header h1 {
    text-indent: -9999px;          /* 隐藏 h1 中的文本 */
    width: 691px;   height: 104px;
    background: #fff url(images/logo.jpg) no-repeat 64px 0;
                                   /* logo 图像左侧有 64px 空白 */
    margin: 0 0 0 161px;           /* 向右移动 161px */    }
```

提示：将标题中的文字进行图像替换最主要的目的就是在 HTML 代码中仍然保留 h1 元素中的文字信息，这样对于网页的维护和结构完整都有很大好处，同时对搜索引擎的优化也有很大的意义，因为搜索引擎对 h1 标题中的信息相当重视。

2. 网页主体部分的分栏

(1) 页面主体部分可分为＃nav 和＃content 两栏，可将这两栏均设置为浮动，以便让它们并列排列，但问题是两栏可能不等高，需要用其他办法让它们看起来等高。解决办法是在两栏外添加一个容器＃container，结构代码如下：

```
<div id="container">
    <div id="nav">…</div>
    <div id="content">…</div>
</div>
```

(2) 设置整个容器＃container 的背景色为绿色，设置右边栏＃content 的背景色为白色，这样＃content 的白色覆盖在＃container 的右边，＃container 的左侧栏就是绿色了，看起来左右两列就等高了。另外设置＃content 右边框为 1px 实线，作为网页的右边框。

```
#container {
    background-color:#9c0;
    width:852px; }
#container #content {
    width:690px;
    background-color: White;
    float:left;
    border-right: #daeda3 1px solid;        /* 网页主体部分的右边框 */}
```

3. 制作左侧列导航块

(1) 设置左侧列中的导航块样式。由于在表格布局中导航块宽度是 161px，而里面导航项的宽度是 143px，所以可以设置＃nav 块的 width 为 152px，左填充为 9px，这样＃nav 的宽度就有 161px，而它里面的导航项左右也正好有 9px 的宽度，实现水平居中。

```
#container #nav {
    float:left;
    width:152px;   height:166px;
    background-color:#00801b;
    padding:15px 0 0 9px;      }
```

(2) 在＃nav 块中添加 6 个 a 元素作为导航项，HTML 代码如下：

```
<div id="nav">
    <a href="#">首 页</a><a href="#">关于我们</a><a href="#">产品与服务</a>
    <a href="#">新闻中心</a><a href="#">职业发展</a><a href="#">联系我们
</a>
```

```
</div>
```

（3）然后设置这些导航项的样式，其中导航项的背景图如图 4-102 所示，设置导航项在默认状态下显示该背景图的上部，鼠标滑过时显示下部即实现了背景翻转效果。

```
#nav a {
    display:block;
    width:113px; height:18px;
    background:url(images/dh.jpg) no-repeat;
    padding:5px 0 0 30px;
    color:white; text-decoration:none;
    font:12px/1.1 "黑体";   }
#nav a:hover {
    color:#00801b;
    background-position:0 -23px; }
```



图 4-102　a 元素导航项的背景图

提示：如果要将图像作为元素的背景显示在网页中，只需设置元素的宽和高等于图像的宽和高即可，但如果对元素还设置了填充值，就必须将元素的宽和高减去填充值。例如，a 元素的背景图尺寸是 143×23px，但由于设置了填充值，因此对 a 元素的宽和高设置为 113px 和 18px。

（4）但是当#container 里的两列都浮动后，它们都脱离了标准流，此时#container 不会容纳它们(IE 除外)，必须在它里面放置一个清除浮动的元素用来扩展#container 的高度。

```
<div id="container">
    <div id="nav"></div>
    <div id="content"></div>
    <div id="clear"></div>
</div>
#container #clear {    clear:both;    }
```

当然，也可以设置#container 元素(overflow:auto)来清除浮动的影响。

4. 制作右侧主要内容栏

（1）接下来设置页面主体的内容部分#content，可发现#content 盒子里包含三个子盒子，分别用来放置上方的 banner 图片、中间的公司简介栏目和底部的版权信息，因此在元素#content 中插入三个子 div 元素。代码如下：

```
<div id="content">
    <div id="banner"></div>
    <div id="intro">  …  </div>
    <div id="copyright">  …  </div>
</div>
```

（2）设置#banner 盒子的宽和高正好等于 banner 图片(ba1.jpg)的宽和高，再设置

#banner 的背景图是 banner 图片，就完成了 banner 区域的样式设置。代码如下：

```
#content #banner {
    background: url(images/ba1.jpg) no-repeat;
    width:688px; height:181px;        /* 宽和高正好等于 ba1.jpg 的大小 */}
```

(3) 设置公司简介栏目#intro，可发现公司简介栏目由标题和内容两部分组成，因此在其中插入两个 div。由于标题#title 部分有两个背景图像，需要两个盒子，所以在#title 里面再添加一个 h2 元素。代码如下：

```
<div id="intro">
    <div id="title"><h2>公司简介</h2></div>
    <div id="intr">光普太阳能成立于…<img src="images/in.jpg"/>…</div>
</div>
```

(4) 接下来设置#title 的样式，由于#title 上方和左边需要留一些空隙，因此设置其 margin 属性和 width 属性使其水平居中，设置其背景图像为一张小背景图像横向平铺。

```
#intro #title {
    width:90%;
    margin:16px 0 0 5%;              /* 设置上边界和左边界,实现水平居中 */
    background:url(images/bj.jpg) repeat-x;        /* 背景图横向平铺 */}
```

(5) 再对 h2 设置背景图像，因为需要对 h2 元素进行图像替代文本，设置 h2 的高度把#title 盒子撑开，再设置 margin 为 0 消除 h2 的默认边界距。

```
#intro #title h2 {
    text-indent:-9999px;             /* 隐藏 h2 的文本 */
    background:url(images/ggd.jpg) no-repeat;
    height:41px;
    margin:0;     }
```

(6) 设置公司简介栏目文本的样式，主要是设置边界、字体大小、行高、字体颜色等。

```
#content #intro #intr {
    width:90%;
    margin:21px 0 0 5%;              /* 设置上边界和左边界,实现水平居中 */
    font-size: 9pt; line-height: 18pt; color: #999;     }
```

再设置文本区域中的客服人员图片右浮动，实现图文混排。

```
#intro #intr img {                   /* 文本里的客服人员图像 */
    float:right;                     /* 右浮动,实现图文混排 */
    width:300px; height:200px;       /* 宽和高正好等于 in.jpg 的大小 */}
```

(7) 设置网页底部版权部分样式，包括用上边框制作一条水平线和设置文本样式。

```
#content #copyright {
    font-size: 9pt; color: #999; text-align:center;
```

```
    width:90%;
    margin:8px 0 0 5%;   padding:8px;
    border-top:1px solid #ccc;     }
```

总结：通过上面的代码可看出，由于要定义每个盒子在网页中的精确大小，几乎每个元素的盒子都设置了 width 和 height 属性，只是有些父元素可以被子元素撑开，所以父元素的这些属性有时可以省略。

为了让元素的盒子在网页中精确定位，一般可通过元素自身的 margin 和父元素的 padding 属性使盒子精确移动到某个位置，像＃header 中的 h1 元素就是通过 margin 属性移动到了右侧。

4.8.4 可变宽度布局

随着显示屏的变大，可变宽度布局目前正变得流行起来，它比固定宽度布局有更高的技术含量。本节介绍三种最常用的可变宽度布局模式，即：两列（或多列）等比例布局，一列固定、一列变宽的 1-2-1 式布局，两侧列固定、中间列变宽的 1-3-1 式布局。

1. 两列（或多列）等比例布局

两列（或多列）等比例布局的实现方法很简单，将固定宽度布局中每列的宽由固定的值改为百分比就行了。

```
#header,#pagefooter,#container{
    margin:0 auto;
    width:85%;              /* 改为比例宽度 */   }
#content{
    float:right;
    width:66%;              /* 改为比例宽度 */   }
#side{
    float:left;
    width:33%;              /* 改为比例宽度 */   }
```

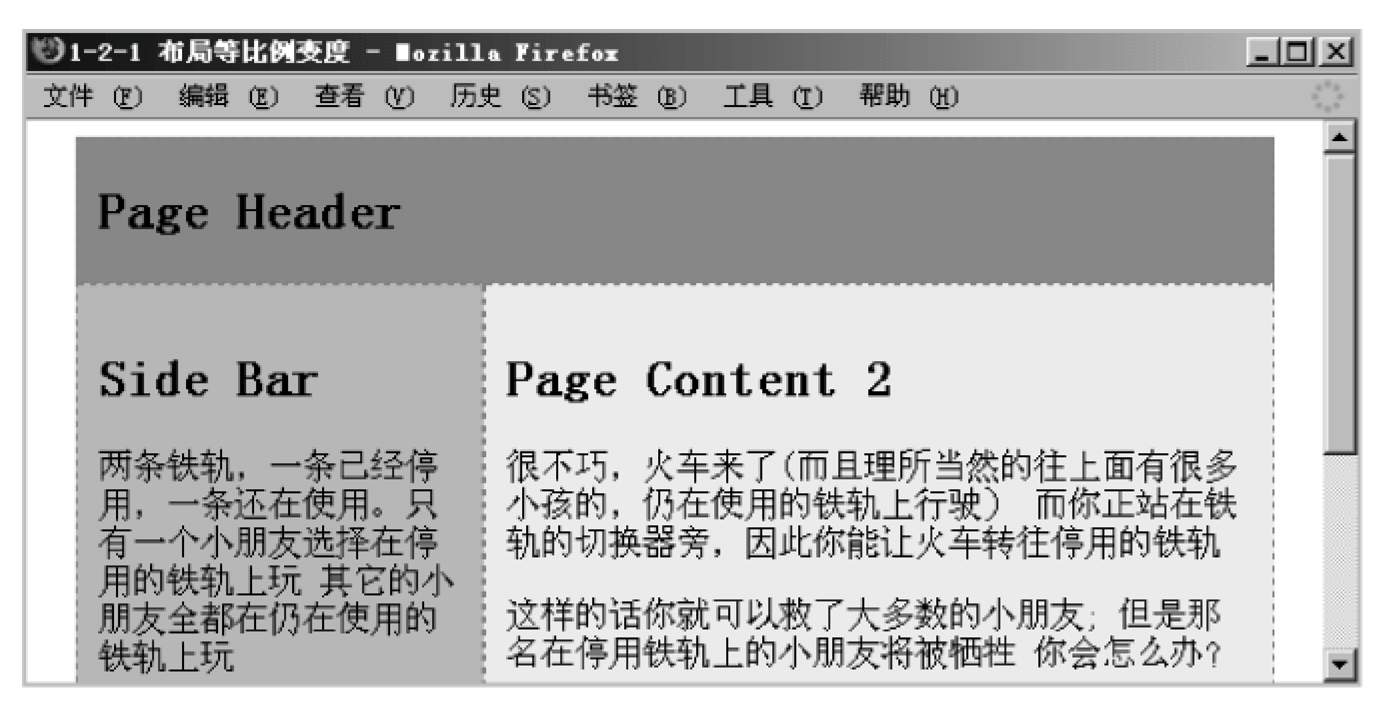
这样不论浏览器窗口的宽度怎样变化，两列的宽度总是等比例的。如图 4-103①所示。

但是当浏览器变得很窄之后，如图 4-103③所示，网页会变得很难看。如果不希望这样，可以对＃container 添加一条 CSS 2.1 里面的“min-width：490px；”属性，即网页的最小宽度是 490px，这样对于支持该属性的 IE 7 或 Firefox 来说，当浏览器的宽度小于 490px 后，网页就不会再变小了，而是在浏览器的下方出现水平滚动条。

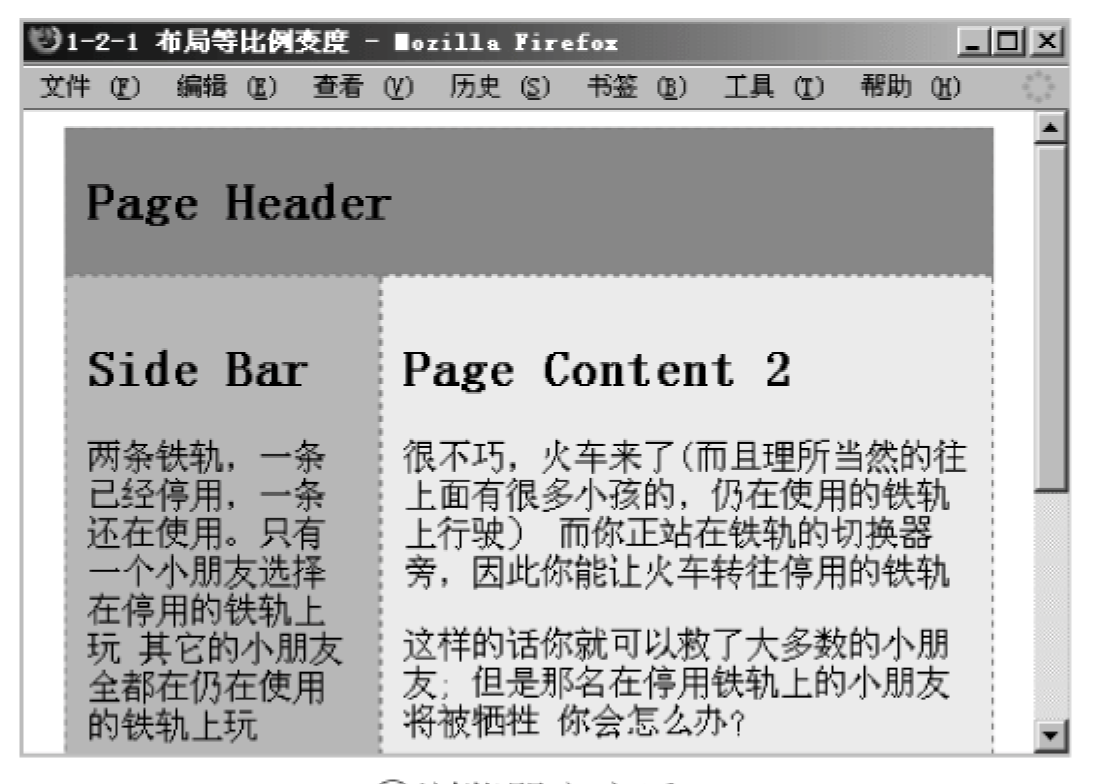
2. 单列变宽布局——改进浮动法

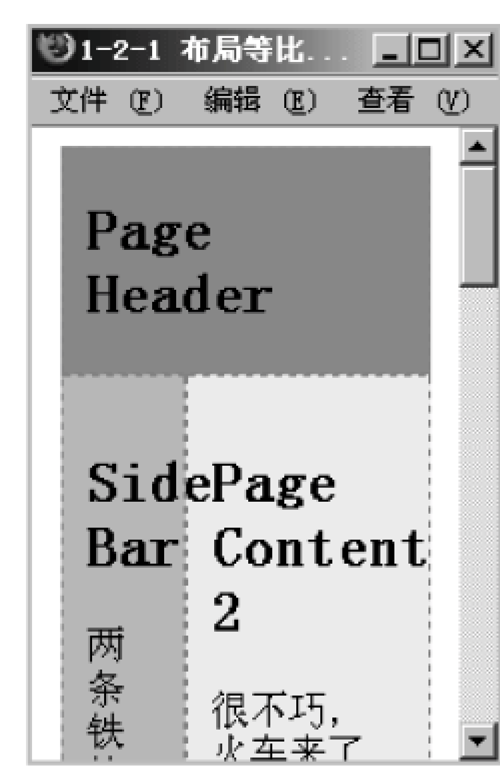
一列固定、一列变宽的 1-2-1 式布局是一种在博客类网站中很受欢迎的布局形式，这类网站常把侧边的导航栏宽度固定，而主体的内容栏宽度是可变的，如图 4-104 所示。

例如，网页的宽度是浏览器宽度的 85％，其中一列的宽度是固定值 200px。如果用表



①浏览器比较宽时





②浏览器变窄后　　③浏览器变得很窄之后

图 4-103　等比例变宽布局时在浏览器窗口变化时的不同效果

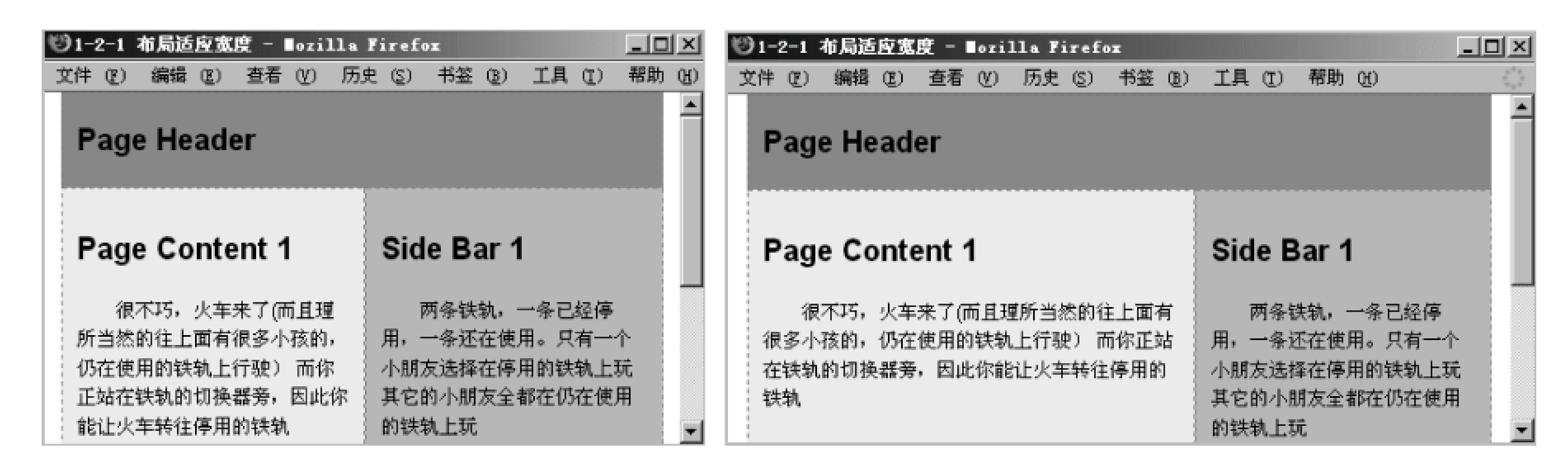


图 4-104　一列固定，一列变宽布局(右边这一列宽度是固定的)

格实现这种布局，只需把布局表格的宽度设为 85%，把其中一列的宽度设为固定值就可以了。但用 CSS 实现一列固定、一列变宽的布局，就要麻烦一些。首先，我们把一列 div 的宽度设置为 200px，那么另一列的宽就是：包含整个网页 container 宽的“100% － 200px”，而这个宽度不能直接写，因此必须设置另一列的宽是 100%，这样另一列就和 container 等宽，这时会占满整个网页，再把这一列通过负边界 margin-left：－200px 向左偏移 200px，使它的右边留出 200px，正好放置 side 列。最后设置这一列的左填充为 200px，这样它的内容就不会显示到网页的外边去。代码如下，图 4-105 是该布局方法的示意图。

```
#header,#pagefooter,#container{
```

```
        margin:0 auto;
        width:85%;    }
    #contentWrap{
        margin-left:-200px;
        float:left;
        width:100%;    }
    #content{
        padding-left:200px;    }
    #side{
        float:right;
        width:200px;    }
    #pagefooter{
        clear:both;    }
    <div id="header">…</div>
    <div id="container">
        <div id="contentWrap">
            <div id="content">…</div>
        </div>
        <div id="side">…</div>
    </div>
    <div id="pagefooter">…</div>
```

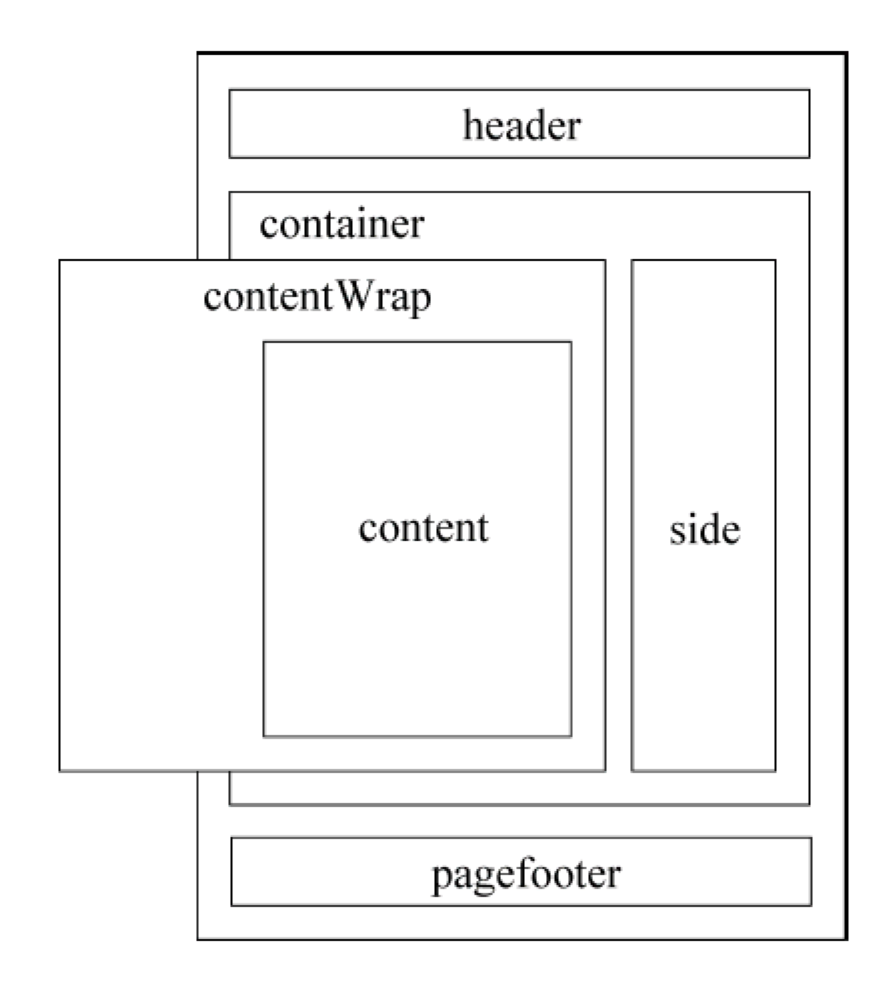


图 4-105 单列变宽布局—改进浮动法示意图

3. 1-3-1 中间列变宽布局——绝对定位法

两侧列固定、中间列变宽的 1-3-1 式布局也是一种常用的布局形式，这种形式的布局通常是把两侧列设置成绝对定位元素，并对它们设置固定宽度。例如左右两列都设置成 200px 宽，而中间列不设置宽度，并设置它的左右 margin 都为 200px，使它不被两侧列所遮盖。这样它就会随着网页宽度的改变而改变，因而被形象地称为液态布局。其结构代码和 1-3-1 固定宽度布局一样，代码如下：

```
<div id="header"><h2>Page Header</h2></div>
<div id="container">
    <div id="navi"><h2>Navi Bar</h2>…</div>
    <div id="content"><h2>Page Content</h2>…</div>
    <div id="side"><h2>Side Bar</h2>…</div>
</div>
<div id="footer"><h2>Page Footer</h2></div>
```

然后将 container 设置为相对定位，则两侧列以它为定位基准。如果此时对两侧列的盒子设置背景色，那么两侧列就可能和中间列不等高，如图 4-106 所示，这样很不美观。

因此不能对两侧列设置背景色，而应该对 container 设置背景色，再对中间列设置另一种背景颜色，这样两侧列的背景色实际上就是 container 的背景颜色了，而中间列的背景色覆盖在 container 的背景色上面，这样两侧列看起来就和中间列等高了，实现代码如下。

```
#navi,#side {
    width: 200px;
    position: absolute;              /* 两侧列绝对定位 */
    top: 0;      }
#navi {
    left:0;                          /* 从#container的左侧开始定位 */     }
#side {
    right:0;                         /* 从#container的右侧开始定位 */     }
#content {
    background:white;                /* 设置中间列背景色为白色 */
    margin: 0 200px;                 /* 不占据两侧列的位置 */     }
#container {
    width:85%;
    margin:0 auto;                   /* 网页居中 */
    background-color:orchid;         /* 设置容器的背景色为淡紫色 */
    position: relative;              /* #container作为左右两列的定位基准 */     }
```


图 4-106　基本的 1-3-1 中间列变宽布局

上面的方法两侧列的背景颜色总是相同的。如果希望两侧列的背景颜色不同，则需要在 container 里面再套一层 innerContainer，给 container 设置一个背景图片从左边开始垂直平铺，给 innerContainer 设置一个背景图片，从右边开始平铺。两个背景图片都只有两列宽，不会覆盖对方。它的结构代码如下，效果如图 4-107 所示。

```
<div id="container">
<div id="innerContainer">
    <div id="navi">…</div>
        <div id="content">…</div>
        <div id="side">…</div>
    </div>
</div>
```

其 CSS 代码就是在上述基础上修改了＃container 选择器并新建了＃innerContainer 选择器，代码如下：

```
#container {
    width:85%;     margin:0 auto;     /* 水平居中 * /
    background:url(images/bg-right.gif) repeat-y right top;
    position: relative;               /* #container 作为左右两列的定位基准 * /      }
#innerContainer {
    background:url(images/bg-left.gif) repeat-y left top;      }
```

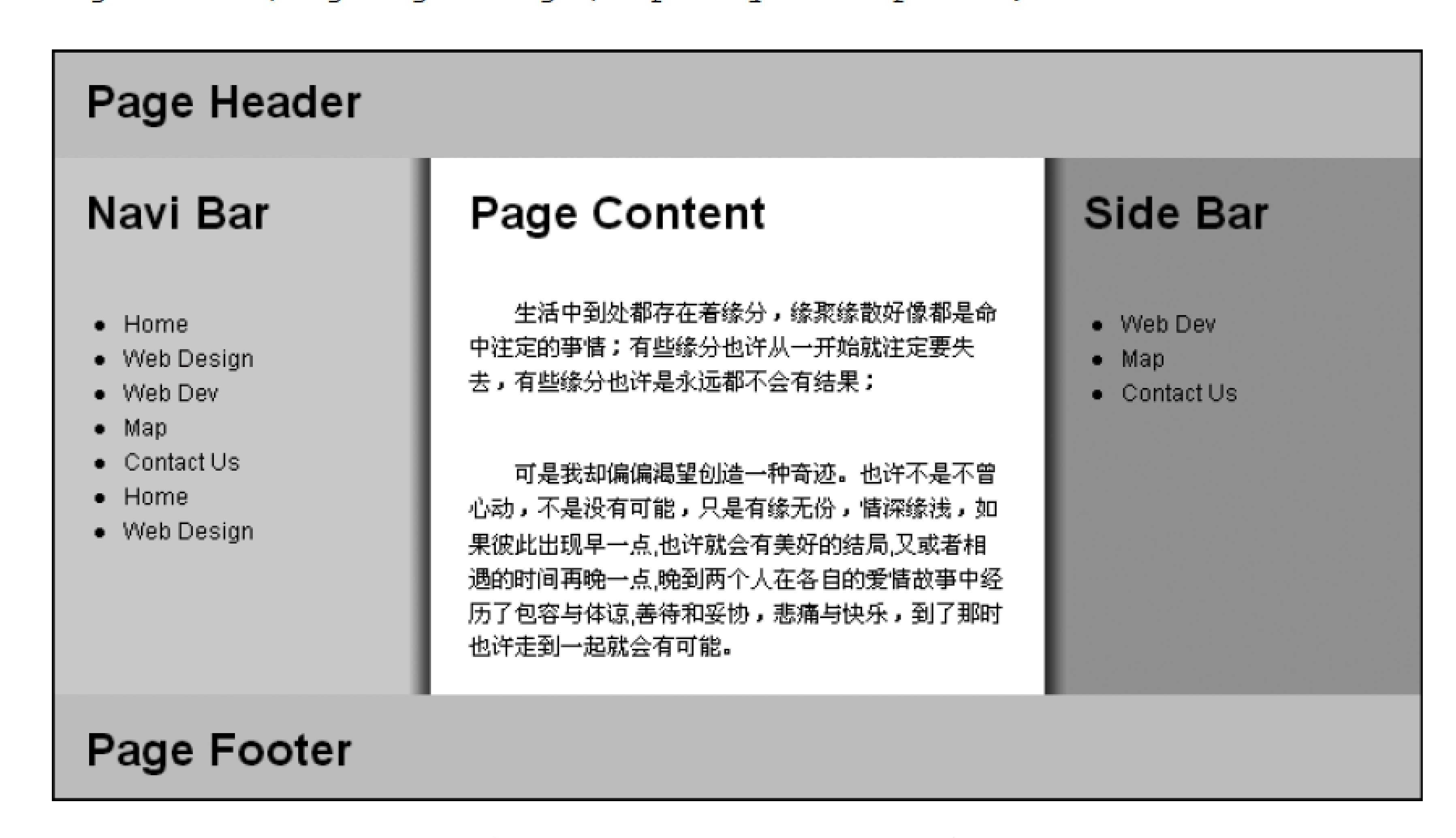


图 4-107　使用带阴影的两个背景图像的页面效果

可以看到，使用这种方式还可以给两个背景色图片设置一些图像效果，例如图 4-107 中的内侧阴影效果。

4.8.5　HTML 5 新增的文档结构标记

在 CSS+DIV 布局中，通常给网页的每个区域的 DIV 都设置一个 ID 属性，属性值一般是 header、footer、nav、sidebar 等。例如，下面是一个 1-2-1 布局网页的结构代码：

```
<body>
<div id="header">页头</div>
<div id="nav">导航</div>
<div id="container">
    <div id="siderbar">左侧栏</div>
    <div id="main">主栏</div>
</div>
<div id="footer">底部说明</div>
</body>
```

尽管上述代码不存在任何错误，还可以在 HTML 5 环境中很好地运行，但该页面结构对于浏览器来说都是未知的，因为元素的 ID 值允许开发者自己定义，只要开发者不同，那么元素的 ID 值就可能各异。

为了让浏览器更好地理解页面结构，HTML 5 中新增了一些页面结构标记。这些新标记可明确地标明页面元素的含义，如头部<header>、导航<nav>、脚部<footer>、分区<section>、文章<article>等。将上述代码修改成 HTML 5 支持的页面代码，如下所示。

```
<body>
    <header>页头</header>
    <nav>导航</nav>
    <section>
        <aside>左侧栏</aside>
        <article>主栏</article>
    </section>
    <footer>底部说明</footer>
</body>
```

这样就可直接对上述结构标记设置 CSS 样式，代码如下，在支持 HTML 5 的浏览器中显示效果如图 4-108 所示。

```
<style type="text/css">
header, nav, section,article,aside,footer{
    border:solid 1px #666;
    padding:10px;
    margin:4px auto;    }
section{padding:4px 0;}
header,nav,footer{ width:400px }
section{width:420px;margin:6px auto;}
aside {
    float:left;
    width:60px;
    height:100px;
    margin:4px 4px 4px 0;    }
article{
    float:left;
    width:312px;
    height:100px;   }
footer {
    clear:both;     }
</style>
```

可见，在 HTML 5 中，使用 CSS 布局已经不再需要 div 了。也不再需要自己设置布局元素的 ID 属性，从标准的元素名就可以推断出各个部分的含义。这对于音频浏览器、手机浏览器和其他非标准浏览器尤其重要。

其中，<article>元素常用来创建一个栏目框，在该元素中还可以有自己的独立元素，如<header>或<footer 等>，示例如下。这样不仅使内容区域各自分段、便于维护，



图 4-108　HTML 5 标记布局的网页

而且代码简单,局部修改也更方便。

```
<article>
<header>
    <h3>HTML 5</h3>
</header>
<p>HTML 5是下一代 HTML 的标准,目前仍然处于发展阶段。经过了 Web 2.0时代,基于互联网的应用
已经越来越丰富,同时也对互联网应用提出了更高的要求。</p>
<footer><p>发表于 2014.10.18</p></footer>
</article>
```

4.9　CSS 浏览器的兼容问题*

由于 CSS 样式以及页面各种元素在不同浏览器中的表现不同,所以必须考虑网页代码的浏览器兼容问题。要解决兼容性问题,一般遵循以下两个原则:

(1) 尽量使用兼容属性,因为并不是所有的 CSS 属性都存在兼容的问题,所以如果使用所有浏览器都能理解一致的属性,那么兼容的问题也就不存在了。

(2) 使用 CSS hack 技术,CSS hack 技术是通过被某些浏览器支持而其他浏览器不支持的语句,使一个 CSS 样式能够按开发者的目的被特定浏览器解释或者不能被特定浏览器解释。

下面介绍几种 CSS hack 的常用技术,它们是针对 IE 6 及以上浏览器和 Firefox 等标准浏览器兼容问题的。

1. 使用!important 关键字

前面已经介绍过!important 关键字的用途,如果在同一个选择器中定义了两条相冲突的规则(注意是在同一个选择器中),那么 IE 6 总是以后一条为准,不认!important ,而 Firefox/IE 7 以定义了!important 的为准。例如:

```
(1) .shadow div {
      background: url(images/top-left.png) no-repeat !important;
                                                    /* Firefox、IE 7执行这一条 */
      background: url(images/top-left.gif) no-repeat;     /* IE 6执行这一条 */
      padding: 0 6px 6px 0;
    }
(2) div{margin:30px !important;      /* Firefox、IE 7执行这一条 */
    margin:28px;                     /* IE 6执行这一条 */
    }
```

2. 在属性前添加“+”、“_”号兼容不同浏览器

在属性前添加“+”号可区别IE与其他浏览器，例如：

```
#demo div{
    width:50px;            /* FireFox 有效 */
    +width:60px;           /* IE 有效 */    }
```

那么如何进一步区分IE 6和IE 7呢？由于IE 7不支持属性前加下划线“_”的写法，会将整条样式忽略，而IE 6却不会忽略，和没加下划线的属性同样解释。例如：

```
#demo div{
    height:50px;           /* FireFox 有效 */
    +height:60px;          /* IE 7 有效 */
    _height:70px;          /* IE 6 有效 */    }
```

3. 使用子选择器和属性选择器等IE 6不支持的选择器

由于IE 6不支持子选择器和属性选择器等CSS选择器，可以用它们区别IE 6和其他浏览器。例如：

```
html body { background-image: (bg.gif) }         /* IE 6 有效 */
html> body { background-image: (bg.png) }       /* FireFox/IE 7有效 */
```

4. 使用IE条件注释

条件注释是IE特有的功能，能够使IE浏览器对XHTML代码进行单独处理。值得注意的是，条件注释是一种HTML的注释，所以只针对HTML，当然也可以将CSS通过行内式方法引入到html中，让CSS也可以应用到条件注释。IE条件注释的使用方法如下：

```
<!--[if IE]>此内容只有 IE可见,其他浏览器会把它当成注释忽略掉<![endif]-->
<!--[if IE 6.0]>此内容只有 IE 6.0可见<![endif]-->
<!--[if IE 7.0]>此内容只有 IE 7.0可见<![endif]-->
```

条件注释也支持感叹号“!”非操作。例如：

```
<!--[if !IE 6.0]>此内容除了 IE 6.0之外都可见<![endif]-->
```

条件注释还可使用 gt 表示 greater than,指当前条件版本的以上版本,不包含当前版本。

gte 表示 greater than or equal,大于或等于当前条件版本,表示当前条件版本和以上的版本。

同样,lt 表示 less than,指当前条件版本的以下版本,不包含当前条件版本。

lte 表示 less than or equal,当前条件版本和它的以下版本。例如:

```
<!--[if lte IE 6]>此内容 IE 6及其以下版本可见<![endif]-->
<!--[if gte IE 7]>此内容 IE 7及其以上版本可见<![endif]-->
```

习　题　4

1. 下列(　　)是定义 CSS 样式规则的正确形式。
   A. body {color=black}　　B. body:color=black
   C. body {color: black}　　D. {body;color:black}
2. 下面(　　)方式不是 CSS 中颜色的表示法。
   A. #ffffff　　B. rgb(255,255,255)
   C. rgb(ff,ff,ff)　　D. white
3. 关于浮动,下列(　　)样式规则是不正确的。
   A. img { float: left; margin: 20px; }
   B. img { float: right; right: 30px; }
   C. img { float: right; width: 120px; height: 80px; }
   D. img { float: left; margin-bottom: 2em; }
4. 关于 CSS 2.1 中的背景属性,下列说法正确的是(　　)。
   A. 可以通过背景相关属性改变背景图片的原始尺寸大小
   B. 不可以对一个元素设置两张背景图片
   C. 不可以对一个元素同时设置背景颜色和背景图片
   D. 在默认情况下背景图片不会平铺,左上角对齐
5. CSS 中定义 .outer {background-color: red;} 表示的是(　　)。
   A. 网页中某一个 ID 为 outer 的元素的背景色是红色的
   B. 网页中含有 class="outer" 元素的背景色是红色的
   C. 网页中元素名为 outer 元素的背景色是红色的
   D. 网页中含有 class=".outer" 元素的背景色是红色的
6. 在图像替代文本技术中,为了隐藏<h1>标记中的文本,同时显示 h1 元素的背景图像,需要使用的 CSS 声明是(　　)。
   A. text-indent: -9999px;　　B. font-size:0;
   C. text-decoration: none;　　D. display: none;

7. 插入的内容大于盒子的边距时，如果要使盒子通过延伸来容纳额外的内容。在溢出(overflow)选项中应选择的是(　　)。

A. visible　　B. scroll　　C. hidden　　D. auto

8. 举例说出3个上下边界(margin)的浏览器默认值不为0的元素________、________、________。

9. CSS中，继承是一种机制，它允许样式不仅可以应用于某个特定的元素，还可以应用于它的________。

10. 如果要使网页中的背景图片不随网页滚动，应设置的CSS声明是________。

11. 设#title{padding：6px 10px 4px}，则ID为title的元素左填充是________。

12. 如果要使下面代码中的文字变红色，则应填入：<h2 ____________>课程资源</h2>

13. 下列各项描述的定位方式是什么？(填写static、relative、aboslute中的一项或多项。)

① 元素以它的包含框为定位基准。________。

② 元素完全脱离了标准流。________。

③ 元素相对于它原来的位置为定位基准。________。

④ 元素在标准流中的位置会被保留。________。

⑤ 元素在标准流中的位置会被其他元素占据。________。

⑥ 能够通过z-index属性改变元素的层叠次序。________。

14. 简述用Dreamweaver新建一条CSS样式规则的过程。

15. 有些网页中，当鼠标滑过时，超链接的下划线是虚线，你认为这是怎么实现的？

16. 为图4-28中的栏目表格添加CSS代码，让栏目中的标题字体为14px、红色、粗体，标题到单元格左边框的距离为12px。让第二行中的文字大小为12px，行距为文字大小的1.6倍，文字与单元格四周边框的距离均为10px，段落前空两格。

17. 简述制作纯CSS下拉菜单的原理和主要步骤。

18. CSS中的display属性和visibility属性都可以用于对网页指定对象的隐藏和显示，它们的效果是一样的吗？请设计一个实验验证。

19. 在HTML中插入一个无序列表和一个有序列表，然后写CSS样式，定义无序列表中内容的字体为14px、黑体，背景色为#fed，水平排列，无项目编号。定义有序列表中的内容的字体是12px，下边框为1px红色虚线，然后使用嵌入式的方法将它插入到HTML代码的正确位置中，再将该CSS代码分别转换为链接式和行内式的形式。

20. 在HTML代码中插入一个a元素，然后用CSS设置它的盒子属性，要求盒子的填充值为(上：6px、下：4px、左右：10px)，边框为1px红色实线，背景为淡红色，并且该盒子能在浏览器中完全显示出来。

第5章 JavaScript

网页有时需要与用户进行交互，例如，响应用户的鼠标键盘操作、验证表单数据以及动态改变 HTML 元素的外观等，这些都需要编写浏览器端程序来实现。JavaScript 是一种浏览器的编程脚本语言，专门用来编写浏览器程序（也称为客户端脚本）。

5.1 JavaScript 的代码结构

JavaScript 是事件驱动的语言。当用户在网页中进行某项操作时，就产生了一个“事件”（event）。事件几乎可以是任何事情：单击一个网页元素、拖动鼠标等均可视为事件。JavaScript 是事件驱动的，当事件发生时，它可以对之做出响应。具体如何响应某个事件由编写的事件处理程序决定。

因此，一个 JavaScript 程序一般由“事件＋事件处理程序”组成。根据事件处理程序所在的位置，在 HTML 代码中嵌入 JavaScript 有三种方式。

1. 将脚本嵌入到 HTML 标记的事件中（行内式）

HTML 标记中可以添加“事件属性”，其属性名是事件名，属性值是 JavaScript 脚本代码。例如（5-1. html）：

```
<html><body>
    <p onclick="alert('Hello,The Web World!');">Click Here</p>
</body></html>
```

其中，onclick 就是一个 JavaScript 事件名，表示单击鼠标事件。“alert(…);”是事件处理代码，作用是弹出一个警告框。因此，当在这个 p 元素上单击时，就会弹出一个警告框，运行效果如图 5-1 所示。



图 5-1　5-1. html 和 5-2. html 的运行效果

2. 使用<script>标记将脚本嵌入到网页中(嵌入式)

如果事件处理程序的代码很长,则一般把事件处理程序写在一个函数(称为事件处理函数)中,然后在事件属性中调用该函数。下面代码的运行效果与5-1.html完全相同。

```
<html><head>              <!--5-2.html -->
<title>第一个 JavaScript 程序</title>
<script>
    function msg () {                          //定义函数 msg
        alert("Hello, the WEB world!");    }
</script></head>
<body>
    <p onclick="msg()">Click Here</p>            <!--通过事件调用函数 -->
</body></html>
```

其中,“onclick="msg()"”表示调用函数msg。可见,调用JavaScript函数可写在HTML标记的事件属性中,但函数的代码必须写在<script></script>标记之间。

将JavaScript代码写成函数的一个好处是,可以让多个HTML元素或不同事件调用同一个函数,从而提高了代码的重用性。

提示:<script>标记是专门用来在HTML中嵌入JavaScript代码的标记。建议将所有的JavaScript代码都写在<script></script>标记之间,而不要写在HTML标记的事件属性内。这可实现HTML代码与JavaScript代码的分离。

3. 使用<script>标记的src属性链接外部脚本文件(链接式)

如果有多个网页文件需要共用一段JavaScript,则可以把这段脚本保存成一个单独的.js文件(JavaScript外部脚本文件的扩展名为js),然后在网页中调用该文件,这样既提高了代码的重用性,也方便了维护,修改脚本时只需单独修改这个js文件的代码。

引用外部脚本文件的方法是使用<script>标记的src属性来指定外部文件的URL。示例代码如下(5-3.html和5-3.js位于同一目录下),运行效果如图5-1所示。

```
------------------------5-3.html的代码------------------------
<html><body>
<script type="text/JavaScript" src="5-4.js "></script>
<p onclick="msg()">Click Here</p>
</body></html>
------------------------5-3.js的代码--------------------------
function msg () {                   //定义函数 msg
    alert("Hello,the WEB world!"); }
```

5.2 JavaScript 的事件编程

编写 JavaScript 程序需要确定三个要素：即触发程序执行的事件、事件处理程序和获取事件作用的 HTML 元素(DOM 对象)。

5.2.1 常用 JavaScript 事件

对于用户而言，常用的 JavaScript 事件可分为鼠标事件、HTML 事件和键盘事件三类，其中常用的鼠标事件如表 5-1 所示，常用的 HTML 事件如表 5-2 所示。

表 5-1 鼠标事件的种类

事 件 名	描 述
onclick	单击鼠标左键时触发
ondbclick	双击鼠标左键时触发
onmousedown	鼠标任意一个按键按下时触发
onmouseup	松开鼠标任意一个按键时触发
onmouseover	鼠标移动到元素上时触发
onmouseout	鼠标移出该元素边界时触发
onmousemove	鼠标指针在某个元素上移动时持续触发

表 5-2 常用的 HTML 事件

事 件 名	描 述
onload	页面完全加载后在 window 对象上触发，图片加载完成后在其上触发
onunload	页面完全卸载后在 window 对象上触发，图片卸载完成后在其上触发
onerror	脚本出错时在 window 对象上触发，图像无法载入时在其上触发
onselect	选择了文本框的某些字符或下拉列表框的某项后触发
onchange	文本框或下拉框内容改变时触发
onsubmit	表单提交时(如单击提交按钮)在表单 form 上触发
onblur	任何元素或窗口失去焦点时触发
onfocus	任何元素或窗口获得焦点时触发
onscroll	浏览器的滚动条滚动时触发

对于某些元素来说，还存在一些特殊的事件，例如 body 元素就有 onresize(当窗口改变大小时触发)和 onscroll(当窗口滚动时触发)这样的特殊事件。

键盘事件相对来说用得较少，主要有 keydown(按下键盘上某个按键触发)、keypress(按下某个按键并且产生了字符时才触发，即忽略 Shift、Alt 等功能键)和 keyup(释放按

键时触发)。通常键盘事件只有在文本框中才显得有实际意义。

提示：JavaScript 事件名应该全部小写，因为 JavaScript 代码是区分大小写的，但是 HTML 标记中的事件属性名却是不区分大小写的。

5.2.2 事件监听程序

实际上，事件除了可写在 HTML 标记中，还可以"对象.事件"的形式出现，这称为事件监听程序。其中对象可以是 DOM 对象、浏览器对象或 JavaScript 内置对象。下面采用事件监听程序的方式重写 5-2.html，代码如下：

```
<html><head>
<script>
    var demo=document.getElementById("demo");
    /* 获取 id为 demo的 HTML元素,由于该 HTML元素的代码在后面,此时尚未载入,会发生"对象不
     存在"的错误 */
    demo.onclick=msg;                  //demo对象单击时执行 msg函数
    function msg()     {
        alert("Hello, the WEB world!");    }
</script></head>
<body>
    <p id="demo">Click Here</p>
</body></html>
```

其中，为 p 元素添加了一个 id 属性，是为了使 JavaScript 脚本方便获取该元素。通过 document.getElementById("demo")方法就可根据 id 访问这个元素，该方法返回的结果是一个 DOM 对象：demo。

然后，通过"DOM 对象.事件名＝函数名"就能设置该对象在事件发生时将执行的函数。

但是，该程序运行会出错，原因在于：浏览器是从上到下依次执行网页代码的。当执行到获取 id 为 demo 的 HTML 元素时，由于该 HTML 元素的代码在下面，浏览器此时尚未载入该元素，就会发生对象不存在的错误。要消除该错误，有以下两种办法。

(1) 把 JavaScript 脚本放在该 HTML 元素代码的下面。修改后的代码如下：

```
<html><body>
<p id="demo">Click Here</p>
<script>
    var demo=document.getElementById("demo");              //将该语句放在 demo元素的后面
    demo.onclick=msg;
function msg()     {
        alert("Hello, the WEB world!");    }
</script>
</body></html>
```

运行该程序，就能得到如图 5-1 所示的运行结果了。

(2) 把获取 HTML 元素的代码写在 windows. onload 事件中,这样就可避免只能把 JavaScript 代码写在 HTML 元素下面的麻烦,其中,windows. onload 事件表示浏览器载入网页完毕时触发,这时所有的 HTML 元素都已经载入到浏览器中,无论将 JavaScript 代码放置到任何位置都不会产生找不到对象的错误。修改后的代码如下:

```
<html><body>
<script>
window.onload=function(){                //表示在网页载入完毕后执行函数
    var demo=document.getElementById("demo");
}
demo.onclick=msg;
function msg()    {
        alert("Hello, the WEB world!");     }
</script>
<p id="demo">Click Here</p>
</body></html>
```

提示:

(1) 程序中的"对象.事件名"后只能接函数名,而绝对不能接函数名加括号。例如,demo. onclick=msg 绝对不能写成 demo. onclick=msg(),因为函数名表示调用函数,而函数名带括号表示运行函数。

(2) demo. onclick=msg; 可放在 window. onload=function(){…}语句外,因为单击事件发生时网页肯定已载入完毕了。

(3) var 是 JavaScript 声明变量的关键词,JavaScript 的变量是一种弱类型变量,任何类型的变量声明都用 var,甚至可以不声明变量直接使用。

(4) JavaScript 代码是严格区分大小写的,每条语句一般以";"号结束。

5.3　JavaScript DOM 编程

5.3.1　动态效果的实现

很多网页中都存在一些动态效果,比如鼠标滑动到某个文本或图像上时,文本或图像会发生变化,或者消失,或者变大变小等,这些都是用 JavaScript 程序实现的。编写动态效果程序的一般步骤是:

(1) 找到要实现动态效果的对象(网页元素);

(2) 为其添加事件;

(3) 编写事件处理函数;

(4) 在事件处理函数中通过改变网页元素的属性或内容来实现动态效果。

下面是一个例子,当鼠标滑动到标题文字上时,标题文字和它下方的图片就会发生变化,效果如图 5-2 所示,代码如下:

```
<html><body>
```

```
<h2 id="tit">会变的图片</h2>
<img src="images/pic1.jpg" id="pic1"/>
<script>                                            //必须写在 pic1 元素后面
var img1=document.getElementById("pic1");          //获取 id为 pic1 的元素
var tit=document.getElementById("tit");            //获取 id为 tit 的元素
tit.onmouseover=change;                            //当鼠标滑动到 tit 元素上时调用 change 函数
function change(){
    img1.src="images/pic2.jpg";                    //设置 img1 的 src 属性为另一张图片
    tit.innerHTML="看到变化了吗";                  //设置 tit 的内容为另一个文本
}
</script>
</body></html>
```





图 5-2　鼠标滑动到标题上文字和图片发生变化的效果

下面分别讲述动态效果程序编写时每一步的实现方法。

5.3.2　获取指定元素

在 JavaScript 中，一般是根据 HTML 元素的 ID 属性或 name 属性或标记名，获取指定的元素，并返回一个 DOM 对象(或数组)。document 对象提供了如下 3 个相关方法。

(1) getElementById()：根据 HTML 元素的 ID 属性访问该元素，并返回一个 DOM 对象。

(2) getElementsByName()：根据 HTML 元素的 name 属性访问该组元素，并返回一个 DOM 对象数组，常用在访问表单元素中。

(3) getElementsByTagName()：根据 HTML 元素的标记名访问该组元素，并返回一个 DOM 对象数组。

其中，getElementById()是最常用的方法，只要给 HTML 元素设置了 ID 属性，就可用该方法访问元素。而由于后面两个方法返回的是数组，所以要使用它们获取单个 HTML 元素，必须添加数组下标，例如：

```
var tj=document.getElementsByName("tj")[1];          //获取第二个 name 属性为 tj 的元素
var mul=document.getElementsByTagName("ul")[0];      //获取第一个<ul>标记的元素
```

1. 添加事件

在获取了要发生交互效果的 HTML 元素后，就可给它添加事件。添加事件可采用 HTML 事件属性或事件监听程序。推荐使用事件监听程序，以实现 HTML 代码与 JavaScript 代码的分离。例如：

```
var tit=document.getElementById("tit");           //获取 id为 tit的元素
tit.onmouseover=change;                  //为 tit元素添加事件,并设置事件处理函数
```

接下来，就可编写事件处理函数，在事件处理函数中，动态效果一般是通过改变 HTML 元素的内容、属性或 CSS 属性实现的。

2. 访问元素的 HTML 属性

当获取到指定的 HTML 元素(DOM 对象)后，就可使用“DOM 对象.属性名”来访问元素的 HTML 属性了。该属性是可读写的，读取和设置元素的 HTML 属性的方法是：

```
变量=DOM对象.属性名                          //读取元素的 HTML属性
DOM对象.属性名=属性值                        //设置元素的 HTML属性
```

下面是一个例子，当鼠标滑动到文字上(p 元素)时，改变该元素的 align 属性，使文字左右跳动，效果如图 5-3 所示，代码如下：





图 5-3 文字左右移动效果

```
<p id="mov" align="left">跳动的文字</p>
<script>                                     //必须写在 mov元素后面
var mov=document.getElementById("mov");        //获取 id为 mov的元素
mov.onmouseover=change;                      //当鼠标滑动到 mov元素上时调用 change函数
function change(){
    if(mov.align=="left"){                   // 读取 mov.align属性并比较
        mov.align="right";}                  // 设置 mov.align属性的值
    else mov.align="left";   }
</script>
```

提示：该例中 mov.align 也可写为 this.align，在 JavaScript 中，如果 this 放置在函数体内，那么 this 指代调用该函数的事件前的对象。

5.3.3 访问元素的 CSS 属性

访问元素的 CSS 属性可以使用“DOM 对象.style.CSS 属性名”的方法。该 CSS 属性也是可读写的，读取和设置元素的 CSS 属性的方法如下：

```
变量=DOM对象.style.CSS属性名            //读取元素的CSS属性
DOM对象.style.CSS属性名=属性值          //设置元素的CSS属性
```

下面是一个例子，当鼠标滑动到文字“沙漠古堡”上时，第一张图片就会变大，同时第二张图片会消失，效果如图5-4所示，代码如下：





图5-4 图片变大或消失的效果

```
<html><body>
<b id="tit">沙漠古堡</b> <b id="tit2">天山冰湖</b><br>
<img src="images/pic1.jpg" id="pic1" width="75"/>
<img src="images/pic2.jpg" id="pic2" width="75"/>
<script>                                          //必须写在HTML元素后面
var pic1=document.getElementById("pic1");         //获取id为pic1的元素
var pic2=document.getElementById("pic2");         //获取id为pic2的元素
var tit=document.getElementById("tit");           //获取id为tit的元素
tit.onmouseover=change;       //当鼠标滑动到tit元素上时调用change函数
function change(){
    pic2.style.display="none";                    //隐藏pic2
    pic1.style.width="140px";                     //设置pic1的宽度值,使它变大
    pic1.style.borderLeft="10px solid red";       //设置pic1的左边框值
}
</script>
</body></html>
```

说明：

（1）CSS样式设置必须符号CSS规范，否则该样式会被忽略。

（2）如果样式属性名称中不带“-”号，例如color，则直接使用style.color就可访问该属性值；如果样式属性名称中带有“-”号，例如font-size，则对应的style对象属性名称为fontSize。转换规则是去掉属性名称中的“-”，再把后面单词的第一个字母大写。又如border-left-style，对应的style对象属性名称为borderLeftStyle。

（3）对于CSS属性float，不能使用style.float访问，因为float是JavaScript的保留字，要访问该CSS属性，在IE中应使用style.styleFloat，在Firefox中应使用style.cssFloat。

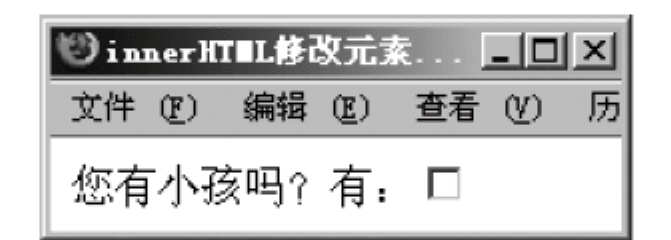
(4) 使用 style 对象只能读取到元素的行内样式，而不能读取元素所有的 CSS 样式。如果将 HTML 元素的 CSS 样式改为嵌入式的话，那么 style 对象是访问不到的。因此 style 对象获取的属性与元素最终显示效果并不一定相同，因为可能还有非行内样式作用于元素。

(5) 如果使用 style 对象设置元素的 CSS 属性，而设置的 CSS 属性和元素原有的任何 CSS 属性冲突，由于 style 会对元素增加一个行内 CSS 样式属性，而行内 CSS 样式的优先级最高，因此通过 style 设置的样式一般为元素的最终样式。

5.3.4 访问元素的内容

如果要访问或设置元素的内容，一般使用 innerHTML 属性。innerHTML 可以将元素的内容(位于起始标记和结束标记之间)改变成其他任何内容(如文本或 HTML 元素)。innerHTML 虽然不是 DOM 标准中定义的属性，但大多数浏览器却都支持它，因此不必担心浏览器兼容问题。

下面是一个例子。当选中表单中的复选框后，将在 span 元素中添加内容(文字和文本框)；若取消选中复选框则清空 span 元素的内容。效果如图 5-5 所示。代码如下：



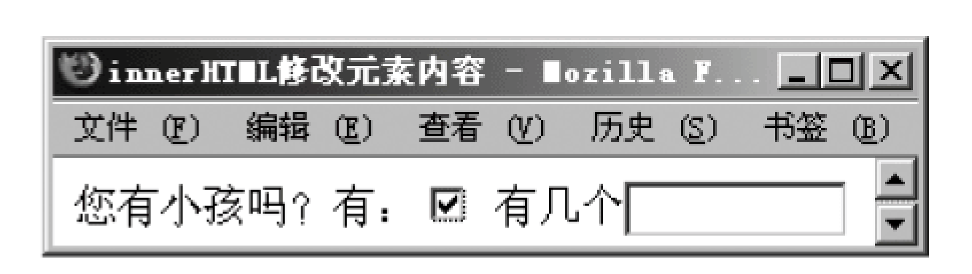


图 5-5 利用 innerHTML 改变元素的内容

```
<form name="userInfo" method="post" action="">您有小孩吗？ 有：
<input type="checkbox" name="hasBoy" id="hasBoy" value="1"
onclick="check()" />
  <span id="add"> </span></form>
<script>
function check(){
    var hasboy=document.forms["userInfo"].hasBoy;
    var add=document.getElementById("add");       //获取 add 元素
    if(hasboy.checked)
        add.innerHTML="有几个<input type='text' name='textfield' />";
    else  add.innerHTML=""; }                    //设置 add 元素的内容
</script>
```

5.4 使用浏览器对象

JavaScript 是运行在浏览器中的，因此提供了一系列对象用于与浏览器进行交互。这些对象主要有 window、document、location、history 和 screen 等，它们统称为 BOM (Browser Object Model，浏览器对象模型)。

window 对象是整个 BOM 的核心，所有其他对象和集合都以某种方式与 window 对

象关联，如图 5-6 所示。

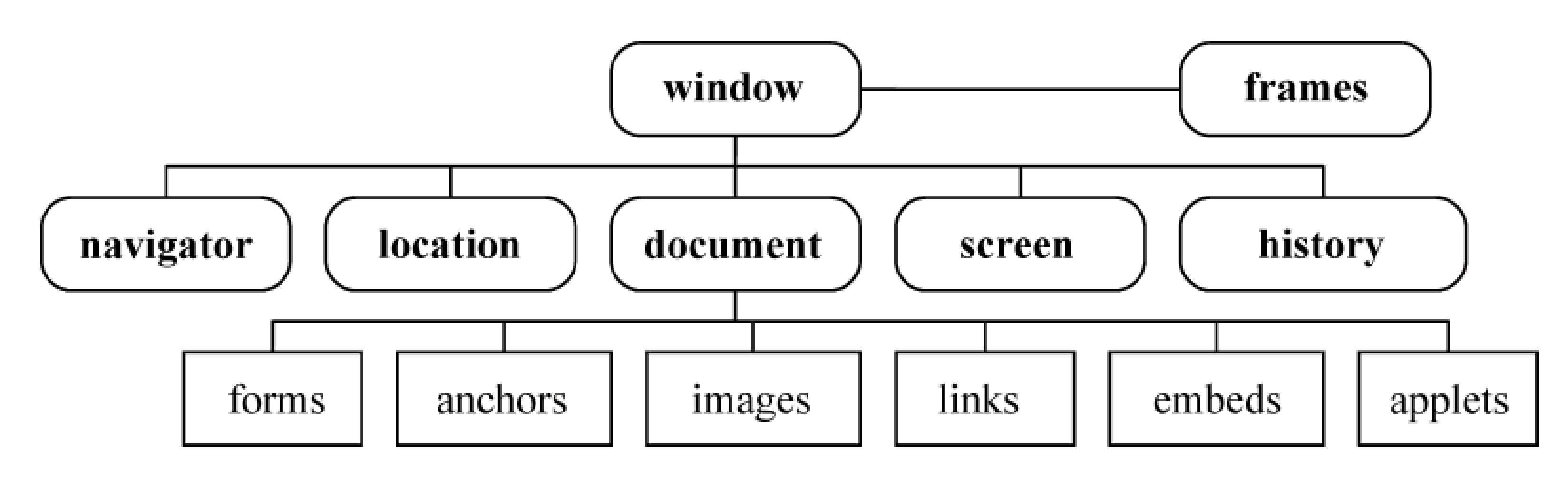


图 5-6　BOM 对象关系图

下面介绍几个最常用对象的含义和用途。

1．document 对象

document 对象表示网页文档，该对象具有很多集合，如 forms、links、images 等，分别表示网页中所有的表单，超链接和图像等集合。因此访问表单和表单中的元素可以使用 forms 集合，例如：

```
document.forms[0].user.value;          //网页第一个表单中 name 属性为 user 元素的 value 值
document.forms["data"].mail;           //名称为 data 的表单中 name 属性为 mail 的元素
```

document 对象还具有 write 方法，用来向网页中动态输出文本，例如：

```
<script> document.write ("这是第一行"+"<br />");</script><!--在网页中输出一行文本-->
```

但要注意的是，document. write 方法只能在文档尚未载入到浏览器时输出文本，如果文档已载入完毕，则 document. write 会清空当前文档内容再输出，例如：

```
<script>
function msg () {
    document.write("Hello!");     }          //输出时会清空原网页内容
</script>
<p onclick="msg()">Click Here</p>
```

由于单击 p 元素时，文档内容已载入完毕，所以单击时，原网页内容会被清空，再输出字符串“Hello!”。因此，document. write 方法不适合于在程序调试时输出中间结果，在 JavaScript 中，这个任务一般是由 alert 方法完成。

2．location 对象

location 对象表示浏览器的 URL 地址，该对象主要用来设置或分析浏览器的 URL，使浏览器发生转向，例如，要使浏览器跳转到 login. htm 页面，代码如下：

```
<script> location.href="login.htm";</script>
```

其中 location. href 是最常用的属性，用于获得或设置窗口的 URL，改变该属性的值就可以导航到新的页面。实际上，Dreamweaver 中的跳转菜单就是采用下拉菜单结合 location. href 属性实现的。下面是一个跳转菜单的代码：

```
<select name="select" onchange="location.href=this.options[this.selectedIndex].value">
    <option>请选择需要的网址</option>
    <option value="http://www.sohu.com">搜狐</option>
    <option value="http://www.sina.com">新浪</option>
</select>
```

如果不希望跳转后用户可以用“后退”按钮返回原来的页面，可以使用 replace()方法，该方法也能转到指定的页面，但不能返回到原来的页面了，这常用在注册成功后禁止用户后退到填写注册资料的页面。例如：

```
<p onclick="location.replace('http://www.sohu.com');">搜狐</p>
```

可以发现，转到新页面后，“后退”按钮是灰色的了。

3. history 对象

history 对象主要用来控制浏览器后退和前进。它可以访问历史页面，但不能获取到历史页面的 URL。下面是 history 对象的一些用法：

```
history.back();                  //浏览器后退一页,等价于 history.go(-1);
history.forward();               //浏览器前进一页,等价于 history.go(1);
history.go(0);                   //刷新当前网页,等价于 location.reload();
document.write(history.length);  //输出浏览历史的记录总数
```

4. window 对象

window 对象对应浏览器的窗口，使用它可以直接对浏览器窗口进行各种操作。window 对象提供的主要功能可以分为 5 类：

（1）调整窗口的大小和位置；

（2）打开新窗口或关闭窗口；

（3）产生系统对话框；

（4）状态栏控制；

（5）定时操作。

下面举几个例子：

```
window.open("pop.html", "new", "width=400, height=300");  //打开一个新窗口
window.moveTo(200, 300);         //移动窗口到指定坐标位置
window.close ();                                           //关闭当前窗口
window.status="看看状态栏中的文字变化了吗?";               //修改状态栏内容
```

5. 系统对话框

window 对象有 3 个生成系统对话框的方法，分别是 alert([msg])、confirm([msg])和 prompt([msg][, default])。由于 window 对象可以省略，因此一般直接写方法名。

（1）alert 方法用于弹出警告框，在框中显示参数 msg 的值，其效果如图 5-7 所示。

（2）confirm 方法用于生成确认提示框，其中包括“确定”和“取消”按钮。当用户单击

“确定”按钮时，该方法将返回 true；单击“取消”按钮时，则返回 false，其效果如图 5-8 所示，代码如下：

```
if(confirm("确实要删除吗?"))                    //弹出确认提示框
    alert("图片正在删除…");
else alert("已取消删除!");
```



图 5-7 确认提示框 confirm()

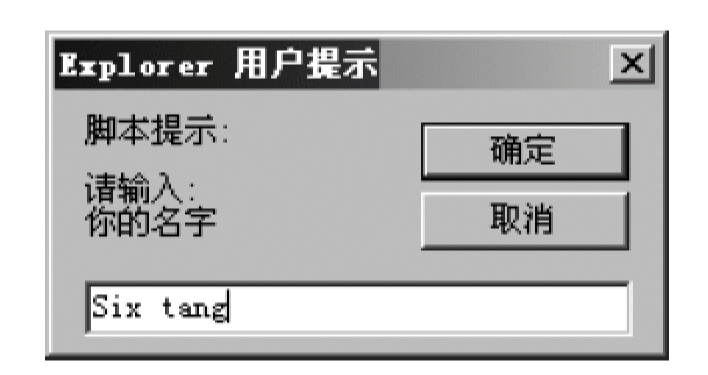


图 5-8 消息提示框 prompt()

（3）prompt 方法用于生成消息提示框，它可接受用户输入的信息，并将该信息作为函数的返回值。该方法接受两个参数：第一个参数是显示给用户的文本，第二个参数为文本框中的默认值（可为空）。其效果如图 5-9 所示，代码如下：

```
var nInput=prompt("请输入:\n你的名字","");         //弹出消息提示框
if(nInput!=null)                                //如果用户输入的值不为空
    document.write("Hello! "+nInput);
```

6. 定时函数

window 对象提供了两个定时函数，分别是 setInterval()和 setTimeout()。定时函数是 JavaScript 制作网页动画效果的基础，例如，网页上的漂浮广告，就是每隔几毫秒更新一下漂浮广告的显示位置。下面分别介绍这两个函数。

（1）setInterval()函数用于每隔一段时间执行指定的代码。需要注意的是，它会创建间隔 ID，若不取消将一直执行，直到页面卸载为止。因此如果不需要了，应使用 clearInterval 取消该函数，以防止它占用不必要的系统资源。

利用 setInterval()周期性地执行显示当前时间的脚本，就可在页面上显示不停走动的时钟，其效果如图 5-9 所示，代码如下：



图 5-9 时钟显示效果

```
<body onload="init()">
    <div id="clock"></div>
<script>
var clock=document.getElementById("clock");       //获取 clock 元素
function disp() {
//将时间显示在 clock 的 div 中,new Date()获取系统时间,并转换为本地格式
    clock.innerHTML="<b>"+ (new Date()).toLocaleString()+"</b>"; }
function init() {
    setInterval(disp, 1000);      // 每隔 1 秒钟执行一次 disp()   }
```

```
</script></body>
```

(2) setTimeout()函数用于在一段时间之后执行指定的代码,这可用于某些需要延时的场合。如果通过递归调用,该函数也能实现周期性地执行脚本。

习　题　5

1. 计算一个数组 x 的长度的语句是(　　)。

A. var aLen=x.length();　　B. var aLen=x.len();

C. var aLen=x.length;　　D. var aLen=x.len;

2. 下列 JavaScript 语句将显示(　　)结果。

```
var a1=10;    var a2=20;    alert("a1+a2="+a1+a2);
```

A. a1+a2=30　　B. a1+a2=1020

C. a1+a2=a1+a2　　D. "a1+a2="1020

3. 表达式"123abc"-123 的计算结果是(　　)。

A. "abc"　　B. 0　　C. -122　　D. NaN

4. 产生当前日期的方法是(　　)。

A. Now();　　B. date();　　C. new Date();　　D. new Now();

5. 下列(　　)可以得到文档对象中的一个元素对象。

A. document.getElementById("元素 id 名")

B. document.getElementByName("元素名")

C. document.getElementByTagName("标记名")

D. 以上都可以

6. 如果要改变元素<div id="userInput">…</div>的背景颜色为蓝色,代码是(　　)。

A. document.getElementById("userInput").style.color="blue";

B. document.getElementById("userInput").style.divColor="blue";

C. document.getElementById("userInput").style.background-color="blue";

D. document.getElementById("userInput").style.backgroundColor="blue";

7. 通过 innerHTML 的方法改变某一 div 元素中的内容(　　)。

A. 只能改变元素中的文字内容　　B. 只能改变元素中的图像内容

C. 只能改变元素中的文字和图像内容　　D. 可以改变元素中的任何内容

8. 下列选项中,(　　)不是网页中的事件。

A. onclick　　B. onmouseover

C. onsubmit　　D. onmouseclick

9. JavaScript 中自定义对象时使用关键字(　　)。

A. Object　　B. Function

C. Define　　D. 以上三种都可以

10. 以下哪条语句不能为对象 obj 定义值为 22 的属性 age?(　　)

A. obj."age"=22;　　B. obj.age=22;

C. obj["age"]=22;　　D. obj={age:22};

11. 下面哪一条语句不能定义函数 f()?(　　)

A. function f(){};　　B. var f=new Function("{}");

C. var f=function(){};　　D. f(){};

12. ________对象表示浏览器的窗口,可用于检索关于该窗口状态的信息。

13. var a=10; var b=20; var c=10; alert(a=b); alert(a==b); alert(a==c); 执行结果是________。

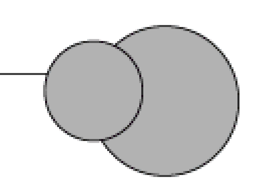
14. 试说明以下代码输出结果的顺序,并解释其原因,最后在浏览器中验证。

```
<script>    setTimeout(function(){ alert("A"); },0);
alert("B");    </script>
```

15. 编写代码实现以下效果:打开一个新窗口,原始大小为 400×300px,然后将窗口逐渐增大到 600×450px,保持窗口的左上角位置不变。

16. 用 JavaScript 编写计算器程序,实现第 9 章图 9-21 中的计算器程序效果。

第6章



PHP动态网站开发概述

目前互联网上绝大多数网站都是动态网站，简单地说，动态网站是一种使用HTTP（超文本传输协议）作为通信协议，通过网络让浏览器与服务器进行通信的计算机程序。制作动态网站可分为两个方面：一是网站界面设计，主要是用HTML、CSS等代码及图片设计网页的界面；二是网站程序设计，实现管理网站内的信息、与用户进行交互等功能。

6.1　动态网站概述

动态网站是一种基于B/S结构的网络程序。那么什么是B/S结构呢？

B/S是Browser/Server的缩写，即浏览器/服务器结构。它是随着Internet技术的兴起，对C/S(Client/Server)客户端/服务器结构的一种变化或者改进。在这种结构下，客户端软件由浏览器来代替，一部分事务逻辑放在浏览器端（Browser）实现，但是主要事务逻辑在服务器端（Server）实现。

动态网站通常由HTML文件、服务器端脚本文件和一些资源文件组成。

（1）HTML文件提供静态的网页内容。

（2）脚本文件提供程序，实现客户端与服务器之间的交互以及访问数据库或文件等。

（3）资源文件提供网站中的图片、样式表及配置文件等资源。

6.1.1　动态网站的运行环境

动态网站的运行，需要Web服务器、浏览器和HTTP通信协议3个要素的支持。

1. Web服务器

动态网站的运行需要一个载体，这就是Web服务器。一个Web服务器可以部署多个网站（或Web应用程序）。

通常Web服务器有两层含义：一方面它代表运行Web应用程序的计算机硬件设备，一台计算机只要安装了操作系统和Web服务器软件，就可算作一台Web服务器；另一方面Web服务器专指一种软件——Web服务器软件，该软件的功能是响应用户通过浏览器提交的HTTP请求，例如用户请求的是PHP脚本，则Web服务器软件将解析并执行PHP脚本，生成HTML格式的文本，并发送到客户端，显示在浏览器中。

2. 浏览器

浏览器是一种用于解析和执行 HTML 代码(可包括 CSS 代码和客户端 JavaScript 脚本)的应用程序,它可以从 Web 服务器接收、解析和显示信息资源(可以是网页或图像等),信息资源一般使用统一资源定位符(URL)标识。

浏览器只能解析和显示 HTML 文件,而无法处理服务器端脚本文件(如 PHP 文件),这就是为什么直接用浏览器能打开 HTML 网页文件,而服务器端脚本文件只有被放置在 Web 服务器上才能被正常浏览的原因。

3. HTTP 通信协议

HTTP 是浏览器与 Web 服务器之间通信的语言。浏览器与服务器之间的会话(见图 6-1)总是由浏览器向服务器发送 HTTP 请求信息开始(如用户输入网址,请求某个网页文件),Web 服务器根据请求返回相应的信息,这称为 HTTP 响应,响应中包含请求的完整状态信息,并在消息体中包含请求的内容(如用户请求的网页文件内容等)。

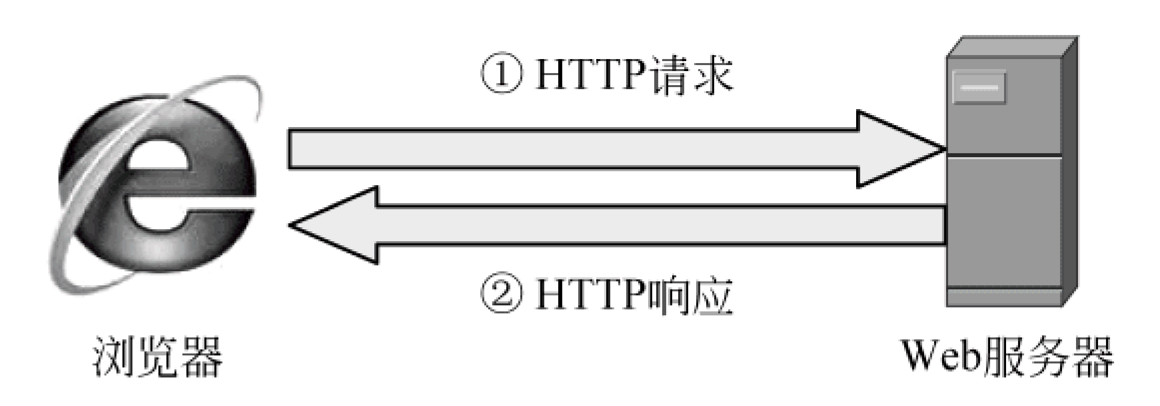


图 6-1 浏览器与服务器之间的会话

6.1.2 动态网站开发语言

动态网站开发语言用来编写动态网站的服务器端程序。目前流行的动态网站开发语言有 CGI、PHP、ASP、ASP.NET 和 JSP 等。下面分别来介绍。

1. CGI

最早能够动态生成 HTML 页面的技术是 CGI(Common Gateway Interface,通用网关接口),由美国 NCSA(National Center for Supercomputing Applications)于 1993 年提出。CGI 技术允许服务器端应用程序根据客户端的请求,动态生成 HTML 页面。早期的 CGI 大多是编译后的可执行程序,其编程语言可以是 C、C++ 等任何通用的程序设计语言,也可以是 Perl、Python 等脚本语言。但是,CGI 程序的编写比较复杂而且效率低,并且每次修改程序后都必须将 CGI 的源程序重新编译成可执行文件。因此目前很少有人使用 CGI 技术。

2. PHP

1994 年,Rasmus Lerdorf 发明了专门用于 Web 服务器编程的 PHP 工具语言,与以往的 CGI 程序不同,PHP 语言将 HTML 代码和 PHP 指令结合成为完整的服务器端动态页面,执行效率比完全生成 HTML 标记的 CGI 要高得多。PHP 的其他优点包括:跨

平台并且开放源代码，支持几乎所有流行的数据库，可以运行在 UNIX、Linux 或 Windows 操作系统下。开发 PHP 时通常搭配 Apache Web 服务器和 MySQL 数据库。

3. ASP

1996 年，Microsoft 公司推出了 ASP 1.0。ASP 是 Active Server Pages 的缩写，即动态服务器页面。它是一种服务器端脚本编程环境，可以混合使用 HTML、服务器端脚本语言（VBScript 或 JavaScript）以及服务器端组件创建动态、交互的 Web 应用程序。从 Windows NT 4.0 开始，所有 Windows 操作系统都提供了 IIS（Internet Information Services）组件，它可以作为 ASP 的 Web 服务器软件。

4. JSP

1997—1998 年，SUN 公司相继推出了 Servlet 技术和 JSP（Java Server Pages）技术。这两者的组合（还可以加上 Javabean 技术），让程序员可以使用 Java 语言开发 Web 应用程序。

JSP 实际上是将 Java 程序片段和 JSP 标记嵌入 HTML 文档中，当客户端访问一个 JSP 网页时，将执行其中的程序片段，然后返回给客户端标准的 HTML 文档。与 ASP 不同的是：客户端每次访问 ASP 文件时，服务器都要对该文件解释执行一遍，再将生成的 HTML 代码发送给客户端。而在 JSP 中，当第一次请求 JSP 文件时，该文件会被编译成 Servlet，再生成 HTML 文档发送给客户端，当以后再次访问该文件时，如果文件没有被修改，就直接执行已经编译生成的 Servlet，然后生成 HTML 文档发送给客户端，由于以后每次都不需要重新编译，因此 JSP 在执行效率和安全性方面有明显优势。JSP 的另一个优点是可以跨平台，缺点是运行环境及 Java 语言都比较复杂，导致学习难度大。

5. ASP.NET

2002 年，Microsoft 公司正式发布了.NET FrameWork 和 Visual Studio .NET，它引入了 ASP.NET 这种全新的 Web 开发技术。ASP.NET 可以使用 VB.NET、C# 等编译型语言，支持 Web 窗体、.NET Server Control 和 ADO.NET 等高级特性。ASP.NET 应用程序最大的特点是程序与页面分离，也就是说，它的程序代码可单独写在一个文件中，而不是嵌入到网页代码中。ASP.NET 需要运行在安装了.Net FrameWork 的 IIS 服务器上。

总的来说，PHP 和 ASP 属于轻量集的 Web 程序开发环境，只要安装了 Dreamweaver 就可进行程序的编写。而 ASP.NET 和 JSP 属于重量级的开发平台，除了安装 Dreamweaver 外，还必须安装 Visual Studio 或 Eclipse 等大型开发软件。

6.1.3　Web 服务器软件

要在计算机上运行动态网站或网页，必须先安装 Web 服务器软件。Web 服务器软件是一种可以运行和管理 Web 应用程序的软件。对于不同的 Web 开发技术来说，其搭配的 Web 服务器软件是不同的。表 6-1 列出了几种 Web 开发语言的特点及其运行

环境。

表 6-1　Web 开发语言的特点及其运行环境

	PHP	ASP	ASP.NET	JSP
Web 服务器	Apache	IIS	IIS	Tomcat
运行方式	解释执行	解释执行	预编译	预编译
跨平台性	任何平台	Windows 平台	Windows 平台	任何平台
文件扩展名	.php	.asp	.aspx	.jsp

Web 服务器的功能是解析 HTTP 协议，当 Web 服务器接收到一个 HTTP 请求后，会返回一个 HTTP 响应，比如返回一个 HTML 静态页面给浏览器、进行页面跳转或者调用其他程序（如 CGI 脚本、PHP 程序等），产生动态响应。而这些服务器端的程序通常会产生一个 HTML 的响应来让浏览器可以浏览。选择 Web 服务器时，应考虑以下因素：性能、安全性、日志和统计、虚拟主机、代理服务器和集成应用程序等。

6.2　网页的类型和工作原理

6.2.1　静态网页和动态网页

根据 Web 服务器是否需要对网页中脚本代码进行解释（或编译）执行，网页可分为静态网页和动态网页。

（1）静态网页：是纯粹的 HTML 页面，网页的内容是固定的、不变的。用户每次访问静态网页时，其显示的内容都是一样的。

（2）动态网页：是指网页中的内容会根据用户请求的不同而发生变化的网页，同一个网页由于每次请求的不同，可显示不同的内容，如图 6-2 中显示的两个网页实际上是同一个动态网页文件（product.php）。动态网页中可以变化的内容称为动态内容，它是由 Web 应用程序来实现的。

在 Internet 发展初期，Web 上的内容都是由静态网页组成，Web 开发就是编写一些简单的 HTML 页面，页面上包含一些文本、图片等信息资源，用户可以通过超链接浏览信息。采用静态网页的网站有很明显的局限性，如不能与用户进行交互，不能实时更新网页上的内容。因此像用户留言、发表评论等功能都无法实现，只能做一些简单的展示型网站。

后来 Web 开始由静态网页向动态网页转变，随着动态网页的出现，用户能与网页进行交互，具体表现在除了能浏览网页内容外，还能改变网页内容（如发表评论）。此时用户既是网站内容的消费者（浏览者），又是网站内容的制造者。

1. 静态网页和动态网页的区别

在 URL 表现形式上，每个静态网页都有一个固定的 URL，而且网页的文件名以 .htm、.html、.shtml 等为后缀，且不含有“?”号。例如，http://ec.hynu.cn/items/g1.



图 6-2　动态网页可根据请求的不同每次显示不同的内容

html。而动态网页的文件名以.php、.asp、.aspx、.jsp、.cgi、.perl 为后缀，文件名后可以有“?”号，例如，http://www.amazon.com/product.aspx?id=204。

在网站的内容上，静态网页内容发布到网站服务器上后，无论是否有用户访问，每个静态网页文件都保存在网站服务器上，每个网页都是一个独立的文件，且内容相对稳定。在网站维护方面，静态网页一般没有数据库的支持，因此在网站制作和维护方面工作量较大，当网站信息量很大时仅仅依靠静态网页来发布网站内容变得非常困难。

提示：动态网页绝不是页面上含有动画的网页，即使在静态网页上有一些动画(如

Flash或Gif动画）或视频，但我们每次访问它时显示的内容是一样的，因此仍然是静态网页。

2. 静态网页的工作流程

用户在浏览静态网页时，Web服务器找到网页就直接把网页文件发送给客户端，服务器不会对网页做任何处理，如图6-3所示。静态网页在每次浏览时，内容都不会发生变化，网页一经编写完成，其显示效果就确定了。如果要改变静态网页的内容，就必须修改网页的源代码再重新上传到服务器。

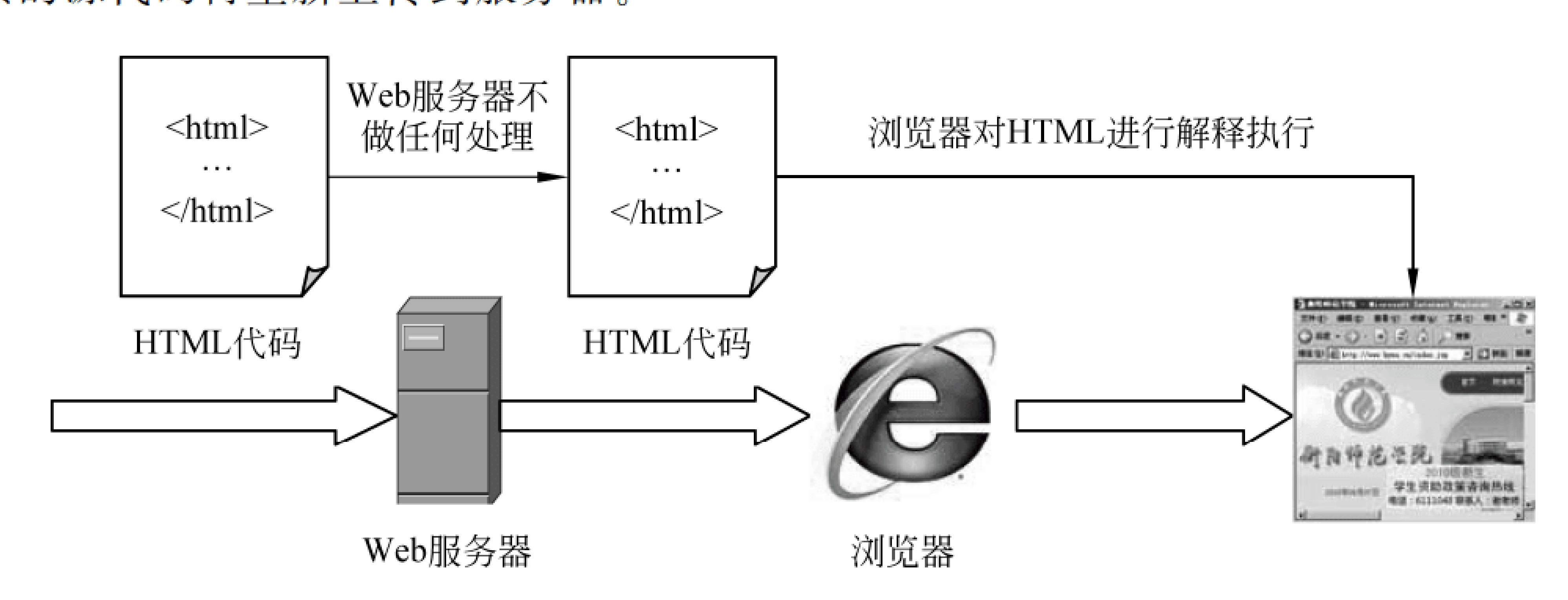


图6-3 静态网页的工作流程

3. 动态网页的优势

静态网页在很多时候是无法满足Web应用的需要的。举个例子来说，假设有个电子商务网站需要展示1 000种商品，其中每个页面显示一种商品。如果用静态网页来做的话，那么需要制作1 000个静态网页，这带来的工作量是非常大的。而且如果以后要修改这些网页的外观风格，就需要一个一个网页地修改，工作量也很大。

而如果使用动态网页来做，只需要制作一个页面，然后把1 000种商品的信息存储在数据库中，页面根据浏览者的请求调用数据库中的数据，即可用同一个网页显示不同商品的信息。要修改网页外观时，也只需修改这一个动态页的外观即可，工作量显著减少。

由此可见，动态网页是页面中内容会根据具体情况发生变化的网页，同一个网页根据每次请求的不同，每次可显示不同的内容。例如一个新闻网站中，单击不同的链接可能都是链接到同一个动态网页，但是该网页每次能显示不同的新闻。

动态网页技术还能实现诸如用户登录、博客、论坛等各种交互功能。动态网页要显示不同的内容，需要数据库作为支持。从网页的源代码看，动态网页中含有服务器端代码，需要先由Web服务器对这些代码进行解释执行，生成HTML代码后才能发送给客户端。

6.2.2 PHP动态网页的工作原理

PHP即Hypertext Preprocessor（超文本预处理器）的递归缩写，是一种服务器端的，跨平台的、开放源代码的多用途脚本语言，尤其适用于Web应用程序开发，并可以嵌入到HTML中去。它最早由Rasmus Lerdorf于1994年发明，而现在PHP的标准是由PHP

Group 和开放源代码社区维护。

PHP 的特点在于跨平台，提供的函数非常丰富，支持广泛的数据库，执行速度快，模板化，能实现代码和页面分离。目前 PHP 的版本为 PHP 5.1(本书介绍这种版本)。

PHP 主要用来编写动态网页，一个完整的动态网页代码可以包含在服务器端运行的程序代码和在浏览器端运行的 HTML 代码。以 PHP 创建的动态网页为例，它的执行过程如图 6-4 所示。

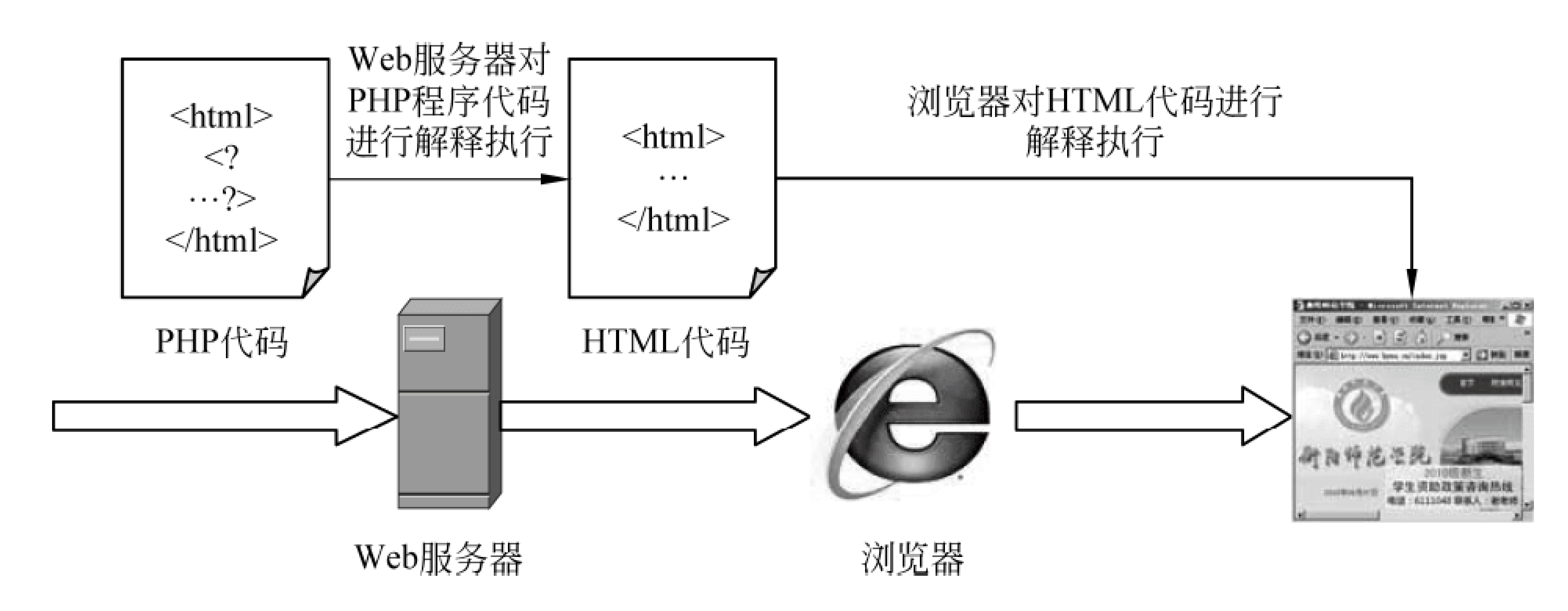


图 6-4　PHP 动态网页的执行过程

可以看出，PHP 程序经过 Web 服务器时，Web 服务器会对它进行解释执行，生成纯客户端的 HTML 代码再发送给浏览器。因此，保存在服务器网站目录中的 PHP 文件和浏览器接收到的 PHP 文件的内容一般是不同的，因此无法通过在浏览器中查看源代码的方式获取 PHP 程序的代码。

图 6-4 中的 Web 服务器主要是指一种软件，它具有解释执行 PHP 代码的功能，PHP 的 Web 服务器软件一般是 Apache。因此，要运行 PHP 程序，必须先安装 Apache，这样才能对 PHP 程序进行解释执行。安装了 Apache 的计算机就成了一台 Web 服务器。

对比一下静态网页，Web 服务器不会对它进行任何处理，直接找到客户端请求的 HTML 文件，发送给浏览器，其运行过程如图 6-3 所示。

因此，Web 服务器的作用是：当浏览器发送请求时，对于静态网页，Web 服务器仅仅是在网站目录中，找到静态网页文件后就直接发送给浏览器；而对于动态网页，Web 服务器找到动态网页后要先对网页中的服务器端代码(如 PHP)进行解释执行，生成纯粹的 HTML 代码后再发送给浏览器。

提示：PHP 文件不能通过双击文件图标直接用浏览器打开，因为这样 PHP 代码没有经过 Web 服务器的处理。运行 PHP 文件的具体方法将在 6.3.2 节介绍。

6.3　配置 PHP 的运行环境

由于 PHP 程序需要在 Web 服务器上运行，因此计算机如果要能运行 PHP 程序，必须在该机上安装 PHP 的 Web 服务器软件——Apache。Apache 有 Windows、Linux 等各种操作系统的版本，这使得 PHP 能运行于不同的操作系统平台上。

对于 PHP 学习者来说，建议采用 Windows＋Apache 2.2＋PHP 5.1＋MySQL 5 作

为 PHP 的运行环境，下面介绍 PHP 的集成环境 AppServ 的安装。

6.3.1 AppServ 的安装

Apache 其实只是一种通用的 Web 服务器，它本身并不能对 PHP 脚本进行解释执行。为了使 Apache 能解释执行 PHP，还须在 Apache 上安装 PHP 的解析器：PHP 5.1。此外，由于开发动态网站通常都需要访问数据库，而 MySQL 是一种很适合与 PHP 搭配使用的数据库，因此，通常还需要安装 MySQL 5。但是 MySQL 是一种没有图形操作界面的数据库管理软件，对数据库的任何操作都只能在命令行下输入命令来完成，这很不友好，为了使 MySQL 能像 Access 那样支持图形界面化操作，又需要安装 MySQL 的图形界面操作程序——phpMyAdmin。

因此，配置 PHP 的运行环境一般需要安装以上 4 种软件。如果分别安装，不仅安装过程很麻烦，而且安装完之后还要进行大量的设置，才能使这几种软件工作在一起。

为此，泰国的 PHP 爱好者制作了 AppServ，AppServ 实际上是这 4 种软件的集成安装包，包含有 Apache、PHP、MySQL、phpMyAdmin。只要安装 AppServ，就可一次性地把 PHP 的运行环境全都安装和配置好，大大简化了 PHP 运行环境的安装和配置。

1. AppServ 的安装过程

AppServ 是一个免费软件，可以在百度上搜索并下载 AppServ，AppServ 的安装文件只有一个，即 appserv-win32-2.5.9.exe，双击该文件，就会弹出安装向导界面，单击 Next 按钮，会出现软件许可协议界面，单击 I Agree 按钮，将提示选择软件的安装位置(见图 6-5)，在这一步的文本框中可直接输入安装路径，建议安装在非系统盘，如 D:\AppServ，并且安装路径中不能含有中文字符，否则会导致错误。



图 6-5 选择安装位置

单击 Next 按钮，选择需要安装的组件(见图 6-6)，因为需要安装 AppServ 包含的 4 种软件，所以，必须把 4 个复选框全部选中。

在下一步中，需要配置 Apache 服务器的有关信息(见图 6-7)，包括服务器名、管理员邮箱和 HTTP 端口号。其中 Server Name 可设置为该服务器的域名，由于我们只是将 Apache 安装在本机上作为测试，因此可以任意输入一个名称。如果是将这台机器作为网

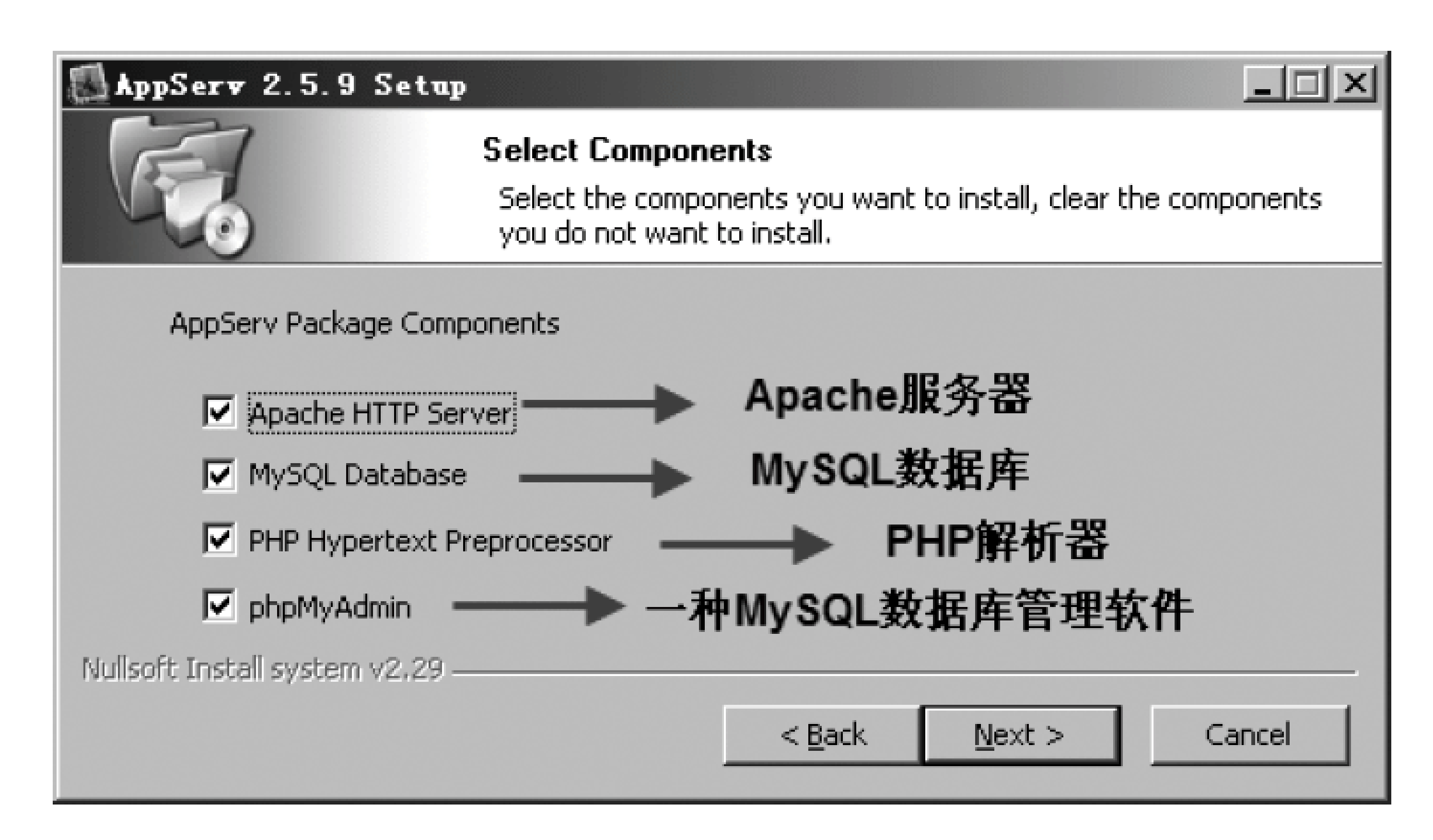


图 6-6　选择安装组件



图 6-7　Apache 服务器信息的配置

络上真正的 Web 服务器，则应该输入一个真实的域名，以便网络上的其他主机都能通过域名访问它。

在 Administrator's Email Address 文本框中，如果填入一个 Email 地址，那么当 Apache 软件运行出现错误时会把错误信息发送到这个邮箱，因此填不填都没关系。

在 Apache HTTP Port 文本框中，可以设置 Apache 服务器 HTTP 服务端口号，建议使用 HTTP 服务默认的 80 端口，如果填其他端口(如 88)，访问时就必须在域名后加上端口号，如 http://localhost:88。

提示：如果本机上还安装了 IIS 或 Tomcat 等其他 Web 服务器，则应该修改这些服务器的 HTTP 端口为非 80 端口，否则，会因为端口冲突，导致 Apache 服务器无法启动。

在下一步中，需要配置 MySQL 数据库的相关信息(见图 6-8)，包括超级用户 root 的密码、MySQL 数据库中字符的编码方式等，本书中设置 MySQL 的 root 用户密码为 111，对于字符编码方式，建议选择默认的 UTF-8 Unicode，因为 UTF-8 编码是世界范围通用的编码方式，它可以保证在英文 IE 浏览器中也能正常显示中文。然后，把下面的 Old Password Support 和 Enable InnoDB 两个复选框都选中，否则可能会导致 MySQL 服务器无法启动。

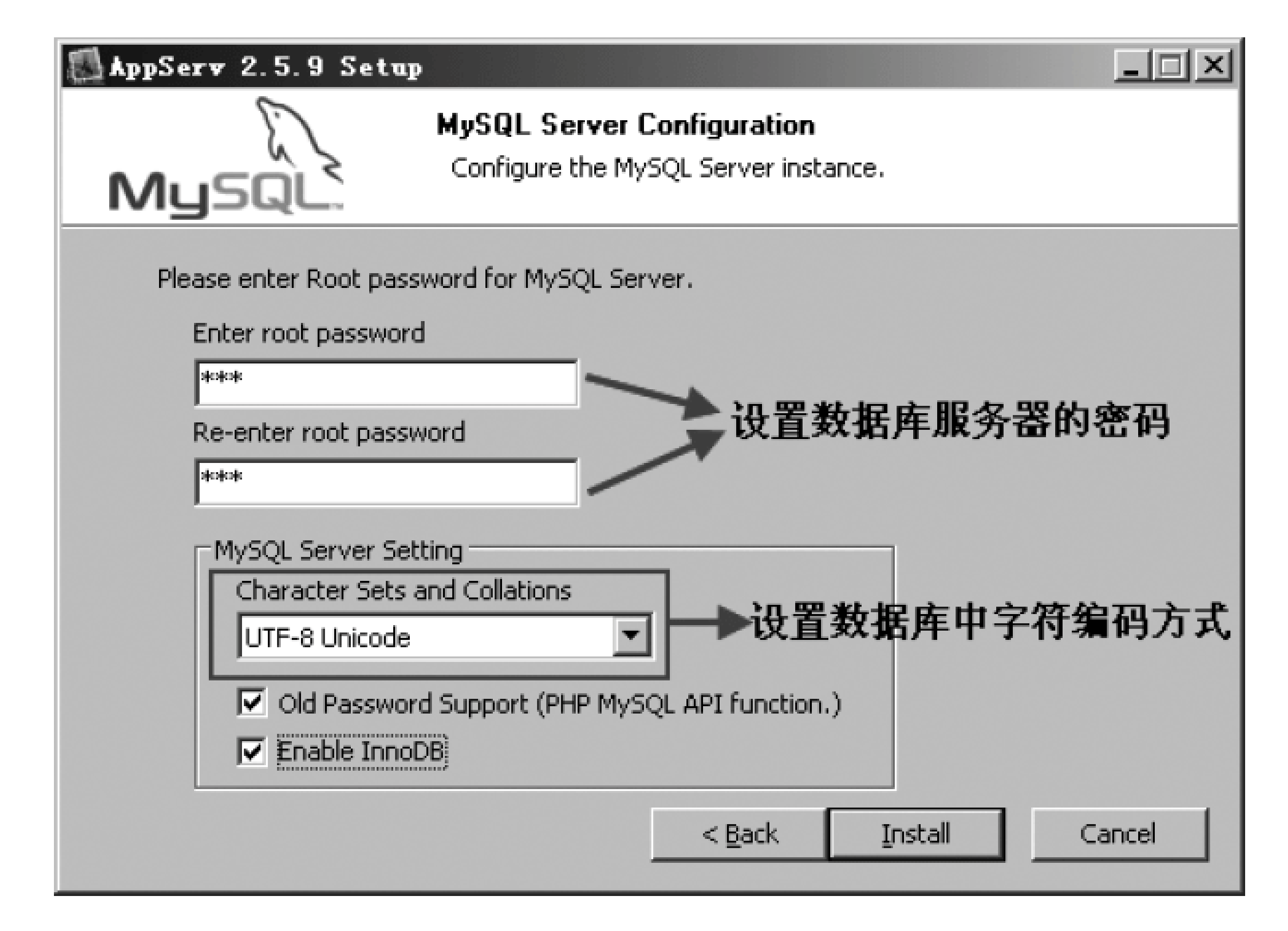


图 6-8 配置 MySQL 数据库服务器

最后单击 Install 按钮，就会开始安装 AppServ 了。安装完成后，默认会自动启动 Apache 服务器和 MySQL 数据库，如果计算机中安装有 360 安全卫士或金山卫士等杀毒软件，可能会提示这些程序正在加入系统服务，这时应该选择“允许”修改。

2. 测试 AppServ 是否安装成功

在 IE 浏览器中输入 http://localhost，如果能看到如图 6-9 所示的网页，则表明 AppServ 安装基本成功了。

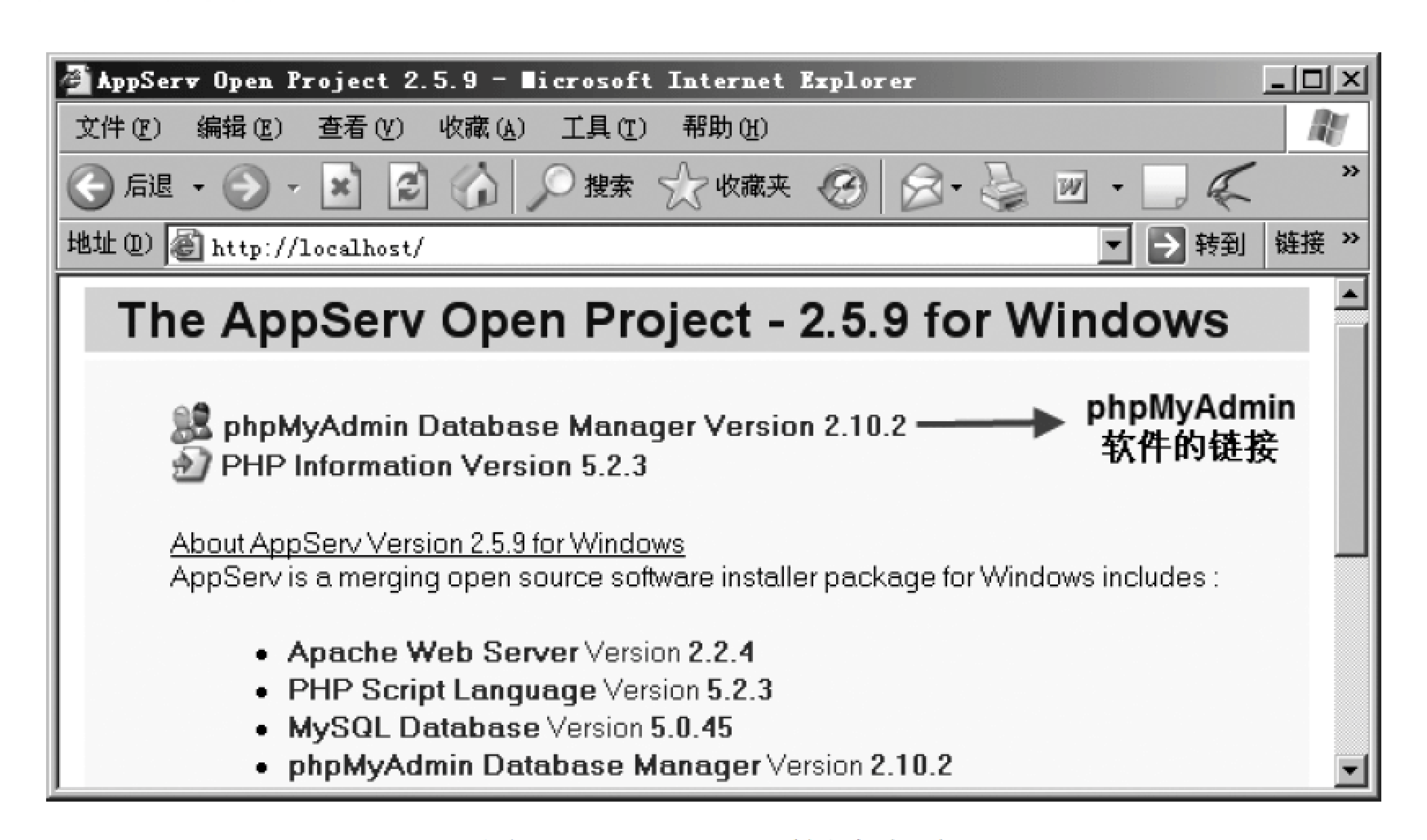


图 6-9 AppServ 的测试页

AppServ 安装完成后，在 AppServ 安装目录下，应该包含 4 个子目录，如图 6-10 所示。其中 Apache2.2 是 Apache 服务器软件的安装目录，MySQL 是 MySQL 数据库软件的目录，php5 是 PHP 解释器的目录，而 www 是网站主目录，双击 www 目录，将进入如图 6-11 所示的目录。

其中，index.php 文件是网站的主页，我们输入 http://localhost 打开的如图 6-9 所示



图 6-10　AppServ 安装目录下的子目录



图 6-11　www 目录下的子目录

AppServ 测试页就是该文件的运行结果。可见，如果要替换 Apache 默认网站的主页，只要替换该文件即可。而 phpMyAdmin 目录是 phpMyAdmin 软件的所在目录。该软件是个 B/S 架构的软件。如果要访问 phpMyAdmin，只要在图 6-9 中单击 phpMyAdmin Database Manager Version 2.10.2 链接，或直接输入网址 http://localhost/phpMyAdmin，就会弹出用户登录框。

在其中输入正确的用户名和密码（这里用户名是 root、密码是 111），就会进入 phpMyAdmin 的界面。在这里可以用图形方式创建和管理数据库及表。phpMyAdmin 的具体使用方法将在 10.1.2 节介绍。

6.3.2　运行第一个 PHP 程序

1. 新建第一个 PHP 程序

PHP 文件和 HTML 文件一样，也是一种纯文本文件，因此可以用记事本来编辑，只要保存成后缀名为.php 的文件就可以了。我们在“记事本”中输入如图 6-12 中所示的代码（注意代码区分大小写）。



图 6-12　在“记事本”中新建一个 PHP 文件

输入完成后，在“记事本”中选择“文件”→“保存”菜单命令，就会弹出如图 6-13 所示的“另存为”对话框，这时首先应在“保存类型”中选择“所有文件”，再在文件名中输入 6-1. php，并选择保存在“D:\AppServ\www”目录下，单击“保存”按钮即新建了一个 PHP 文件 6-1. php。



图 6-13 “另存为”对话框

2. 运行 PHP 文件

PHP 文件只有通过 Web 服务器才能运行，因此刚才将 6-1. php 保存在了 Apache 默认网站的主目录“D:\AppServ\www”下。要运行 Apache 默认网站主目录下的文件，可以在浏览器地址栏中输入如下 URL 访问该文件。

```
http://localhost/6-1.php
```

说明：

(1) http://localhost 相当于本机的域名。我们知道，当在地址栏中输入某个网站的域名后，Web 服务器就会自动到该网站对应的主目录中去找相应的文件。也就是说，域名和网站主目录是一种一一对应的关系，因此 Web 服务器(这里是 Apache)会到本机默认网站的主目录(D:\AppServ\www)中去找文件 6-1. php。

提示：可以通过“http://localhost/文件名”直接访问 Apache 网站根目录下的文件。如果根目录下有子目录的话，要访问子目录下的文件用“http://localhost/子目录名/文件名”即可。例如，假设要访问 D:\AppServ\www\temp 下的 test. php 文件，则在地址栏中输入“http://localhost/temp/test. php”即可。

(2) 关于服务器地址：除了使用 localhost 访问本机外，还可使用 127.0.0.1(本机 IP 地址)访问，这两种方式用来在本机上运行 PHP 文件。第三种方式是使用本机的计算机名访问，这种方式适合在局域网内使用；另外两种方式是使用本机在 Internet 上的域名或 IP 地址访问，这要求本机具有公网上固定的 IP 地址或域名，也就是把本机作为网络上一台真正的 Web 服务器，供 Internet 上其他用户访问该机上的 PHP 文件。

为了简便，本书都采用第一种方式访问。打开浏览器，在地址栏中输入 http://localhost/6-1. php，按回车键，就会出现如图 6-14 所示的运行结果(网页显示的是服务器端的当前日期)。

在图 6-14 中右击，选择“查看源文件”命令，就会出现如图 6-15 所示的源文件，与

图 6-12 中的 PHP 源程序比较，可发现 PHP 代码已经转化成纯 HTML 代码了，这验证了 Web 服务器确实先执行了 PHP 源程序，再将生成的 HTML 代码发送给浏览器。



图 6-14　程序 6-1.php 的运行结果



图 6-15　在浏览器端查看源文件

3. 运行 PHP 程序的步骤总结

PHP 程序需要先经过 Web 服务器(Apache)的解释执行，生成的静态代码才能在浏览器中运行。为了让 Apache 解释执行 PHP 文件，需要经过如图 6-16 所示的两步：

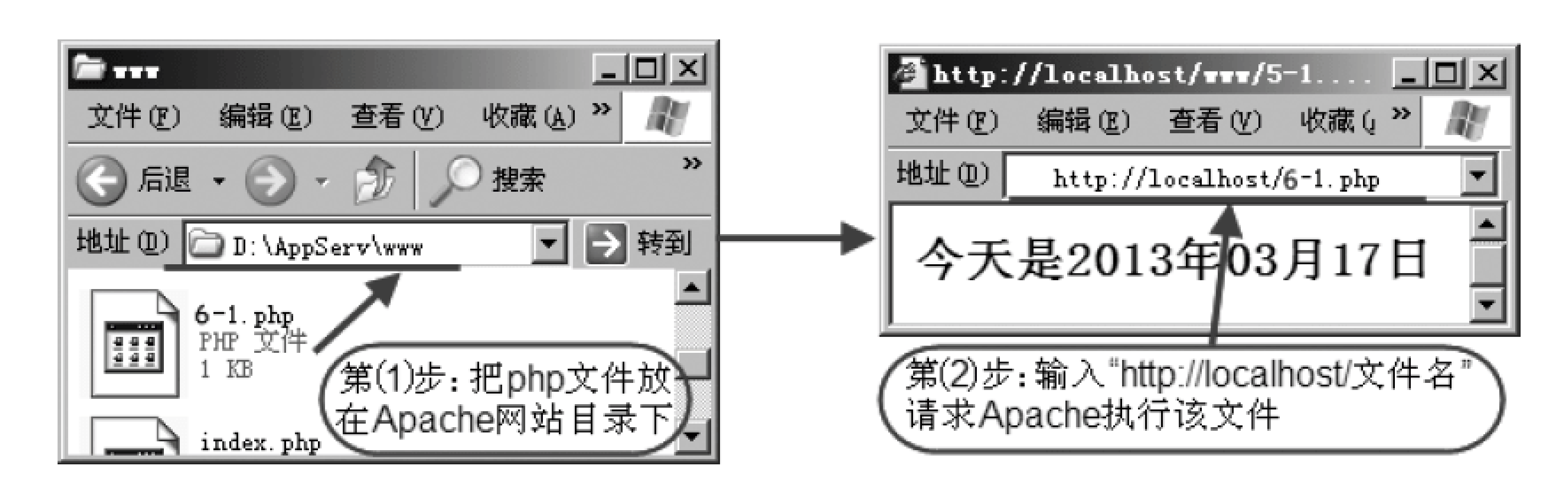


图 6-16　运行 PHP 程序的步骤

(1) 把 PHP 文件放置或保存到 Apache 的网站目录(或子目录)下，这样 Apache 才能找到这个 PHP 文件；

(2) 在浏览器中输入"http://localhost/文件名"，这就相当于向 Apache 服务器发送了一个 HTTP 请求，请求 Apache 执行 URL 地址中的文件，这样 Apache 才会对这个 PHP 文件进行解释执行。

6.3.3　Apache 的配置

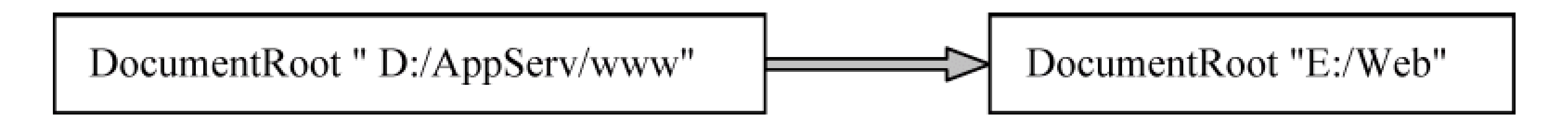
Apache 没有图形化的服务器配置界面，只能通过修改配置文件(httpd.conf)来配置服务器。该文件位于 D:\AppServ\Apache2.2\conf 目录下，对 Apache 服务器的任何设置(如设置主目录、修改首页)都是通过修改该文件的代码来实现的。

httpd.conf 是一个纯文本文件，可以用记事本或 Dreamweaver 等软件打开。也可以在"开始"菜单中，选择"程序"→AppServ→Configuration Server →Apache Edit the httpd.conf Configuration File 命令打开它。

1. 主目录的设置

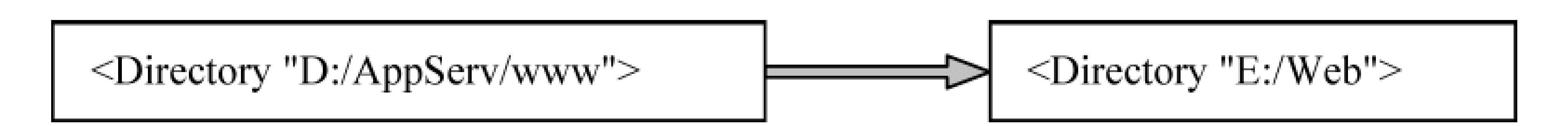
Apache 的网站主目录默认是 D:\AppServ\www，这使得我们要运行 PHP 文件，都必须将它保存在这个目录下，这有些不方便。实际上，可以设置 Apache 的主目录为其他目录，方法如下：

要修改 Apache 的主目录，必须同时修改 httpd. conf 文件中的两个地方。例如，将 Apache 的主目录由 D:\AppServ\www 修改为 E:\Web，则应该先在 E 盘新建文件夹 Web，然后找到 httpd. conf 文件的第 240 行。进行如下修改：



注意：要将目录中的反斜杠改为斜杠，并且目录中不能含有任何中文字符。

再找到 httpd. conf 文件的第 268 行。进行如下修改：



保存文件，重新启动 Apache 服务器，就会将主目录设置成 E:\Web。设置完毕后，可以将文件 6-1. php 从 D:\AppServ\www 目录移动到 E:\Web 目录中，输入 http://localhost/6-1. php 仍然可以访问该文件，因为 http://localhost 对应 E:\Web 目录了。

提示：

(1) 对 httpd. conf 文件进行修改后，都必须重新启动 Apache 才能使设置生效。重启的方法是：在“开始”菜单中，选择“程序”→AppServ→Control Server by Service→Apache Restart 命令；或者在“运行”对话框中输入：httpd-k restart。

(2) httpd. conf 文件中有很多行以“#”开头，“#”其实是注释符，表示这些行都是注释。

(3) 整个 httpd. conf 文件中都不能出现中文或全角字符，否则 Apache 服务器将无法运行。

2. 默认文档的设置

所谓默认文档，就是指网站的主页(首页)，它的作用是这样的，如果在浏览器中只输入 http://localhost 或“http://localhost/子目录名”，并没有输入具体某个网页文件的名称，则 Apache 就会自动按默认文档的顺序在相应的文件夹里查找，找到后就显示。在 httpd. conf 文件的第 303 行中，有对默认文档的设置。

```
#
<IfModule dir_module>
    DirectoryIndex index.php index.html index.htm
</IfModule>
```

可见，在这个设置下，如果不输入具体的网页文件名，则 Apache 首先会在目录下寻找 index. php 文件，如果找不到，就会再去找 index. html 和 index. htm。

默认文档建议保持默认的 index. php 即可，因此此处不用修改。

3. 虚拟目录的建立和访问

如果要在一台计算机的 Apache 上部署(deploy，即建立和运行)多个网站，比如在网上下载了很多个 PHP 网站的源代码想在本机上能同时运行。虽然可以在网站主目录

E:\Web 下建立多个文件夹，每个文件夹下分别放置一个网站的文件。但这样就要把每个网站的文件都移动到网站主目录下的对应目录中，有些麻烦。

更重要的是，由于这些网站都放置在网站主目录下（这就相当于是同一个网站），如果多个网站的程序中都有修改网站公共变量（如同名的 Session 变量）的代码，则可能会发生这个网站修改了其他网站公共变量的情况，从而导致出现意想不到的问题。

设置虚拟目录就是为了解决上述问题的，如果要部署多个网站，可以将一个网站的目录设置为 Apache 的主目录，将其他每个网站的目录都设置为虚拟目录。这样，这些网站都真正独立了，每个网站相当于一个独立的应用程序（Application），它们可以拥有自己的一套公共变量。设置虚拟目录的方法如下：

例如，要新建一个虚拟目录 eshop，指向 E:\eshop 目录，则首先新建 E:\eshop 目录，然后找到 httpd. conf 文件的第 360 行。在＜IfModule alias_module＞后添加一段：

```
Alias "/eshop" "E:\ eshop"
<Directory "E:\eshop">
Options- Indexes FollowSymLinks
AllowOverride None
order allow,deny
Allow from all
</Directory>
```

其中，添加的第一行表示建立了一个虚拟目录 eshop，指向 E:\eshop 目录。而后面一段表示为目录 E:\eshop 设置访问权限。因为在 Apache 中，新建的虚拟目录默认是没有任何访问权限的。重启 Apache 后，虚拟目录 eshop 就建立好了。

提示：Apache 的网站主目录默认是允许目录浏览的，即如果该目录或其子目录下找不到首页文件，而访问者又只输入了域名或目录名，则浏览器会显示该目录下的所有文件和目录列表，这是很不安全的，在 httpd. conf 文件的第 281 行：Options Indexes FollowSymLinks，其中 Indexs 就是表示目录浏览权限，要去掉该权限，可在它前面加个减号“－”，或将其删除，如改成 Options－Indexes FollowSymLinks 即可。

要运行虚拟目录下的文件，可以使用“http://localhost/虚拟目录名/路径名/文件名”的方式访问。比如，在 E:\eshop（对应虚拟目录 eshop）下有一个 index. php 的文件，要运行该文件，只需在地址栏中输入：http://localhost/eshop/index. php 或 http://localhost/eshop。而要运行 E:\eshop\admin 目录下的 index. php 文件，只需在地址栏输入：http://localhost/eshop/admin/index. php，该 URL 的含义如图 6-17 所示。

http://localhost/eshop/admin/index.php

本机域名　虚拟目录名　路径和文件名

图 6-17　访问虚拟目录下文件的 URL

由此可见，访问虚拟目录下文件的 URL 分为三部分，依次是本机域名、虚拟目录名和文件相对于虚拟目录的相对路径和文件名。从访问的 URL 形式上来看，虚拟目录就好像是网站主目录下的一个子目录。

4. 默认端口的修改

如果要修改 Apache 服务器的默认 HTTP 端口，比如将 80 修改为 88，只要找到 httpd. conf 文件的第 67 行。将“Listen 80”改为“Listen 88”即可，以后访问网站主目录就必须使用“域名：端口”的形式（如 http://localhost:88）了。

6.4 使用 Dreamweaver 开发 PHP 程序

Dreamweaver 对开发 PHP 程序有很好的支持，包括代码提示、自动插入 PHP 代码等，Dreamweaver 开发 PHP 程序的最大优势在于，使开发人员能在同一个软件环境中制作静态网页和动态程序。

6.4.1 新建动态站点

开发 PHP 程序之前要先安装和配置好 PHP 的运行环境（Apache），然后就可在 Dreamweaver 中新建动态站点，新建动态站点的作用是：

（1）使站点内的文件能够以相对 URL 的方式进行链接；

（2）在预览动态网页时，能够使用设置好的 URL 运行该动态网页。具体过程如下：

在 Dreamweaver 中执行“站点”→“新建站点”菜单命令，将弹出如图 6-18 所示的新建站点对话框，其中站点名字可任取一个，但是访问该网站的 URL 一定要设置正确。如果该网站所在的目录是 Apache 的网站主目录，则应该用 http://localhost 方式访问，如果该网站所在的目录是 Apache 的虚拟目录，则应该用“http://localhost/虚拟目录名”的方式访问，在这里我们已经把该网站的目录（E:\Web）设置成了 Apache 的主目录，因此在“您的站点的 HTTP 地址（URL）是什么？”下输入“http://localhost/”。

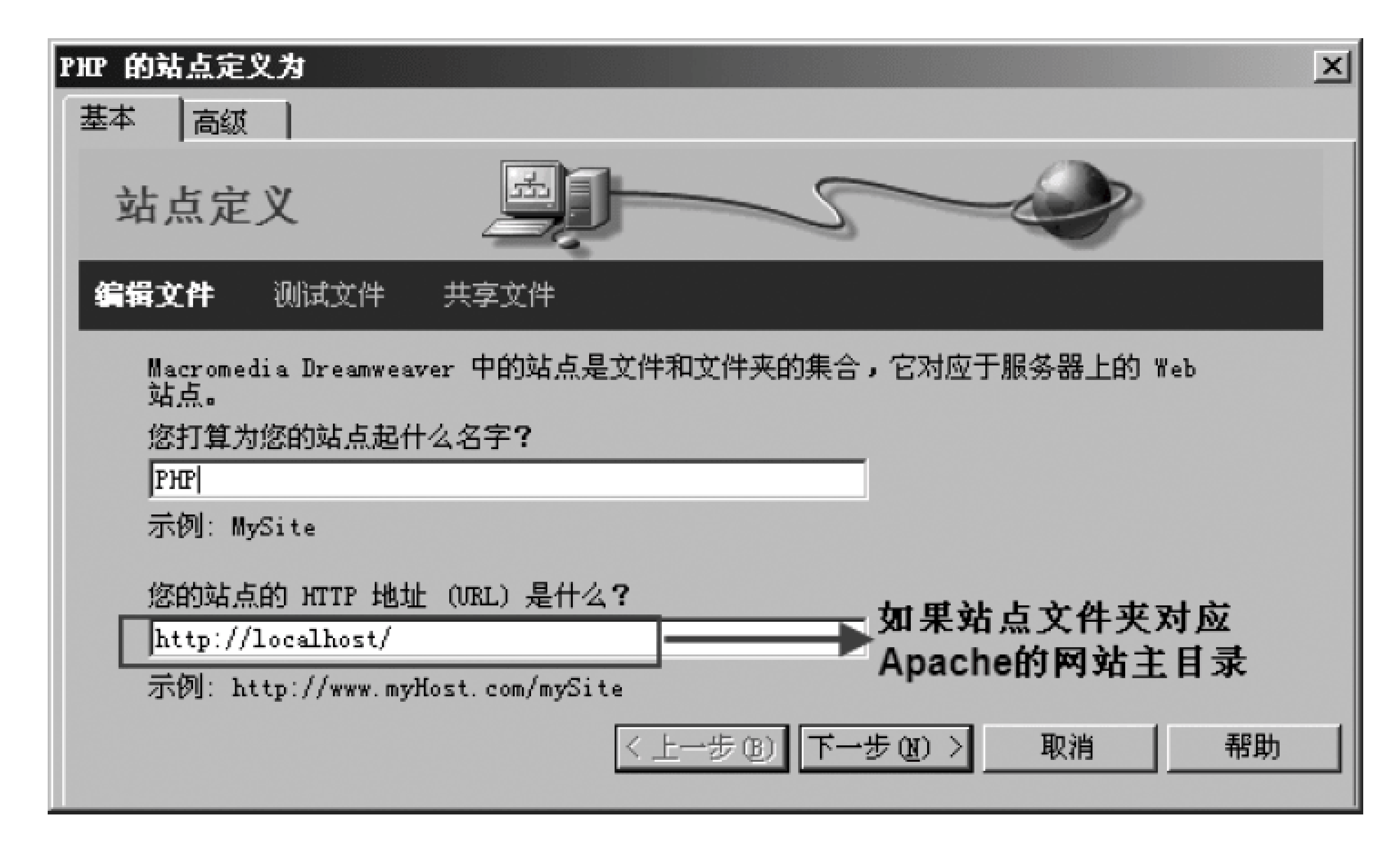


图 6-18 新建动态站点第一步（访问网站的 URL）

单击“下一步”按钮，将出现如图 6-19 所示的对话框，在“您是否打算使用服务器技术……”选项组中，选择“是……”按钮，在“哪种服务器技术”下拉列表框中，选择 PHP MySQL 选项。



图 6-19 新建动态站点第二步(选择服务器技术)

单击“下一步”按钮，在如图 6-20 所示的对话框中先选择“在本地进行编辑和测试”单选按钮。“您将把文件存储在计算机上什么位置?”就是问你的网站的主目录在哪儿，因此必须选择网站的主目录。需要注意的是，该网站的主目录必须和 Apache 的主目录一致，因为 Dreamweaver 预览文件时是打开浏览器并在文件路径前加 http://localhost，这样实际上是定位到了 Apache 的主目录，而不是这里设置的主目录。如果不一致，预览时就会出现“找不到文件”的错误。

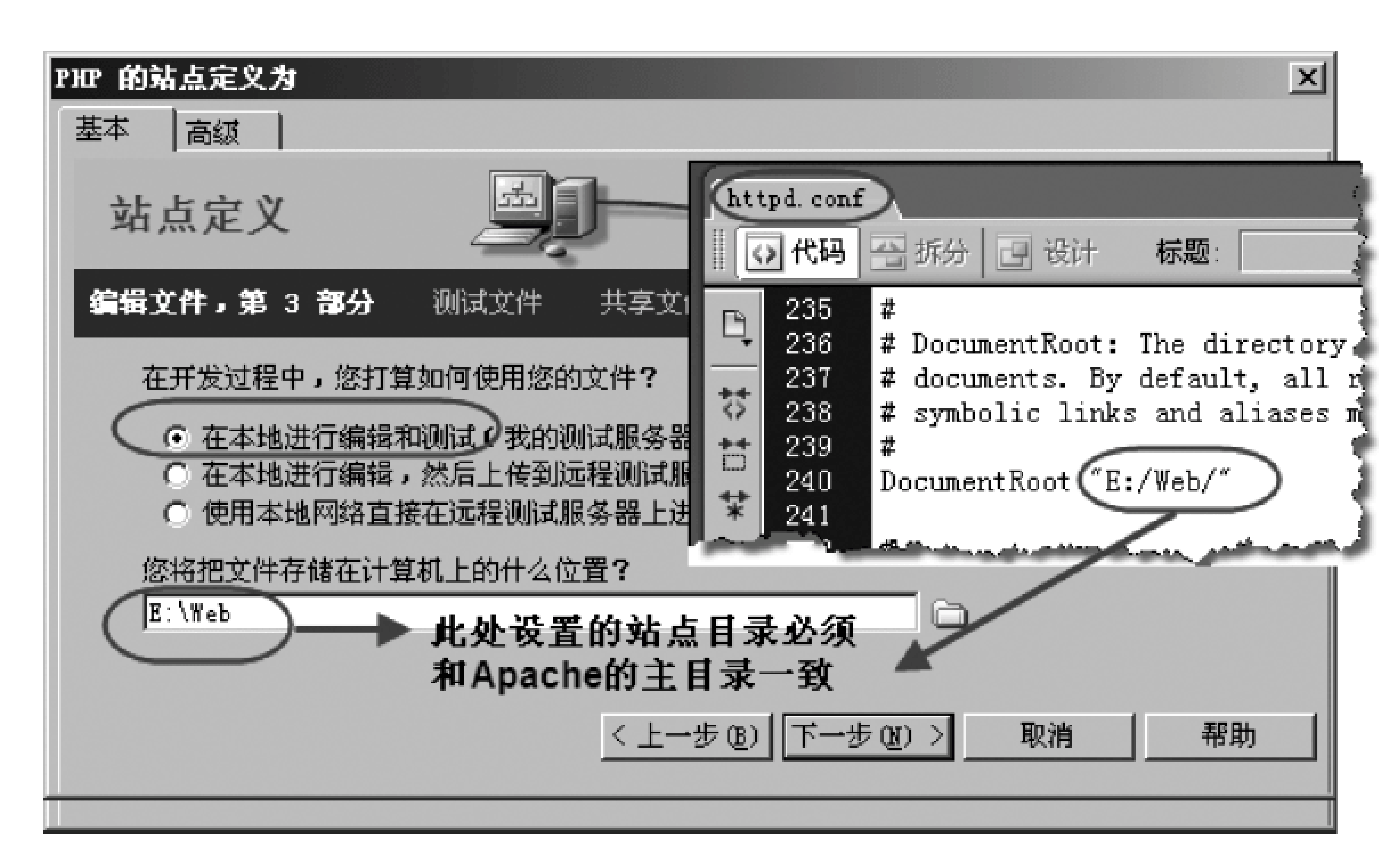


图 6-20 新建动态站点的第三步(设置站点主目录)

单击“下一步”按钮，在图 6-21 所示的对话框中，对于“您应该使用什么 URL 来浏览站点的根目录?”的问题，由于站点根目录是 Apache 的主目录，因此此处仍选择 http://localhost/来浏览根目录。

提示：如果在上一步是选择了“在本地进行编辑和测试”，则这里输入的 URL 应该和图 6-18 中输入的 URL 相同。

最后一步，对于“编辑完一个文件后，是否将该文件复制到另一台计算机中”的问题，选择“否”即可，这样就完成了一个动态站点的建立。

提示：如果网站目录被设置成为 Apache 的一个虚拟目录，如 E:\eshop，则在新建站点时，图 6-18 和图 6-21 中的 URL 应输入：http://localhost/eshop，在图 6-20 中网站目



图 6-21 新建动态站点第四步

录应输入 E:\eshop。

6.4.2 编写并运行 PHP 程序

在 Dreamweaver 中编写并运行 PHP 程序的步骤是：第一，新建 PHP 文件；第二，编写该文件的 PHP 代码；第三，运行 PHP 文件。

1. 新建 PHP 文件

PHP 动态站点建立好之后，可以在 Dreamweaver“文件”菜单中选择“新建”→“动态页”→PHP 命令，就会新建一个 PHP 网页文件(保存时会自动保存为扩展名为.php 的文件)。

或者在如图 1-8 所示的站点“文件”面板中，在站点目录或子目录上右击，选择“新建文件”命令，也能新建一个 PHP 文件。

2. 编写 PHP 代码

在如图 6-22 所示的 Dreamweaver 主界面中，单击“代码”标签，切换到代码视图，就可在其中输入 PHP 代码。如果希望自动插入一些常用的 PHP 代码，可以在工具栏左侧单击选择 PHP，单击工具栏中对应按钮就能将一些常用的 PHP 代码或定界符插入到光标处。

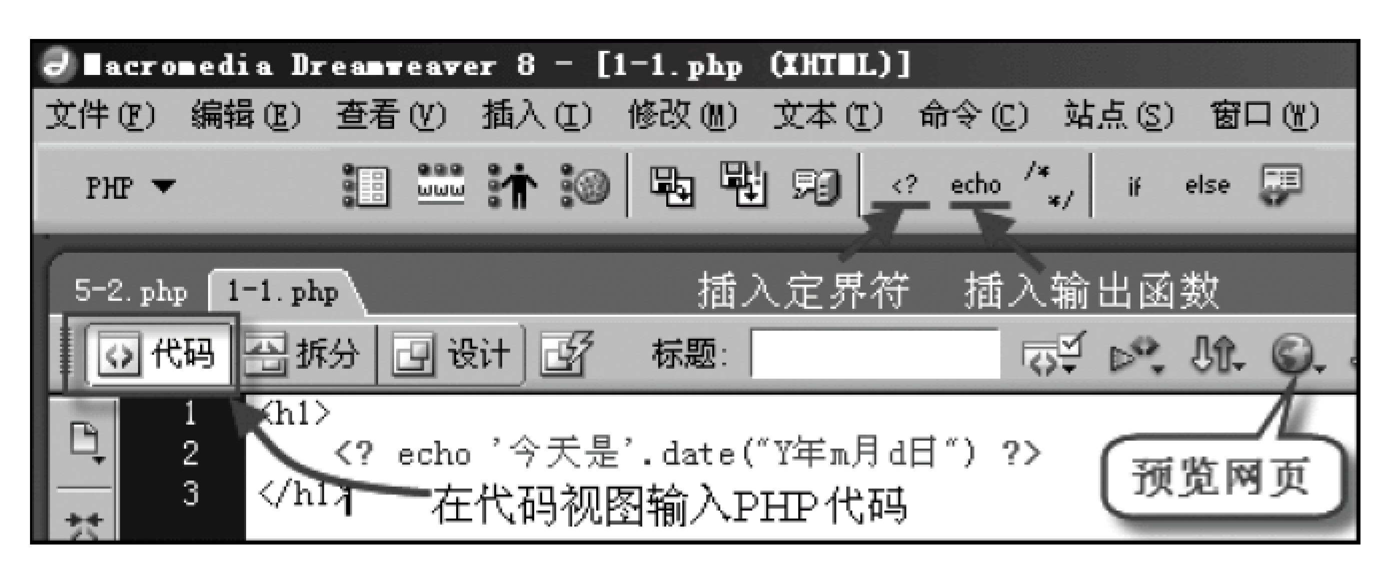


图 6-22 Dreamweaver 中的代码视图

代码编写完毕后,应该按 Ctrl+S 组合键保存文件,第一次保存时会要求输入文件名。

3. 运行 PHP 文件

在如图 6-22 所示的主界面中,单击“预览”按钮(快捷键为 F12),Dreamweaver 将打开浏览器,并自动在浏览器中输入“http://localhost/文件名”来运行该 PHP 文件。

提示:如果预览网页时出现“找不到文件”错误,一般是因为在图 6-20 中设置的站点目录与 Apache 中的网站目录不一致导致的。如果预览网页时浏览器地址栏中没有出现“http://localhost/…”,则是因为新建站点时没有选择使用服务器技术导致的。

习　题　6

1. 以下哪种技术不是服务器端动态网页技术?(　　)。

   A. PHP　　B. JSP　　C. ASP. NET　　D. Ajax

2. 配置 MySQL 服务器时,需要设置一个管理员账号,其名称是(　　)。

   A. admin　　B. root　　C. sa　　D. Administrator

3. 如果 Apache 的网站主目录是 E:\eshop,并且没有建立任何虚拟目录,则在浏览器地址栏中输入 http://localhost/admin/admin. php,将打开的文件是(　　)。

   A. E:\localhost\admin\admin. php

   B. E:\eshop\admin\admin. php

   C. E:\eshop\ admin. php

   D. E:\eshop\localhost\admin\admin. php

4. PHP 的配置文件是________,Apache 的配置文件是________。

5. 如果 Apache 的网站主目录是 E:\eshop,要运行 E:\eshop\abc\rs\123. php 文件,则应在浏览器地址栏中输入________,如果 E:\eshop 是虚拟目录,则要运行 E:\eshop\eshop. php 文件,应在浏览器地址栏中输入________。

6. 对于 Apache 的配置文件,请把左边的项与右边的描述联系起来。

   A. httpd. conf　　(　　)用于设置默认文档;

   B. Listen　　(　　)用于创建虚拟目录;

   C. DocumentRoot　　(　　)用于设置网站的访问端口;

   D. Alias　　(　　)用于设置网站文档的根目录;

   E. DirectoryIndex　　(　　)用于配置 Apache 服务器;

7. Apache 服务器只能支持 PHP 语言吗?

8. 开发 PHP 程序前,使用 Dreamweaver 建立 PHP 动态站点有何作用?

9. 有一个 PHP 文件,存放在 D:\AppServ\www 目录下,请问如果在“我的电脑”中

双击该 PHP 文件，该文件可以运行吗？

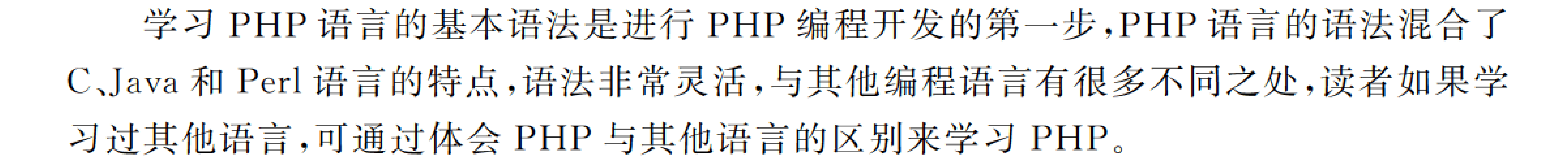
10. 简述动态网站和 Web 应用程序的联系和区别。

11. 列举常见的 Web 服务器软件及动态网页设计语言。

12. 将 Apache 服务器的主目录设置为 D:\wgzx，并运行一个该目录中的 php 文件。

13. 假设已在 Apache 服务器上建立了一个虚拟目录 D:\wgzx，请使用 Dreamweaver 新建一个 PHP 动态站点，站点名称叫 wgzx，该站点目录对应 D:\wgzx 文件夹。

第7章 PHP语言基础



学习PHP语言的基本语法是进行PHP编程开发的第一步，PHP语言的语法混合了C、Java和Perl语言的特点，语法非常灵活，与其他编程语言有很多不同之处，读者如果学习过其他语言，可通过体会PHP与其他语言的区别来学习PHP。

7.1 PHP语法入门

7.1.1 PHP代码的基本格式

PHP是一种可嵌入到HTML中的脚本语言。PHP的代码一般由两部分组成：

(1) HTML代码，还可包括嵌入到其中的CSS和JavaScript代码；

(2) 服务器端脚本，位于PHP定界符“＜?”与“? ＞”之间的代码。

其中，(1)部分是静态网页也具备的，它通过浏览器解释执行，又称为客户端代码。因此，PHP可以通俗地认为是把服务器端脚本放在“＜?”和“? ＞”之间，再嵌入到静态网页中。

提示：“＜?”和“? ＞”称为PHP脚本的定界符，表示脚本的开始和结束。这是因为在PHP文件中，HTML代码和PHP代码混杂在一起(即页面和程序没有分离)，必须使用专门的定界符对PHP代码进行区分。这样服务器就可以只对“＜?”和“? ＞”之间的代码进行执行。

1. PHP代码的4种风格

根据定界符的不同，PHP代码有4种风格，即：XML风格、简短风格、脚本风格和ASP风格。使用任意一种都可将PHP代码嵌入到HTML中去。

1) XML风格

这种风格的PHP定界符是“＜? php ”和“? ＞”(＜? 和php之间不能有空格)。例如：

```
<h1><?php echo '现在是'.date("Y年 m月 d日 H:i:s");?></h1>
```

2) 简短风格

将定界符“＜?php ”中的php省略，就成了简短风格，它的定界符是“＜? ”和“? ＞”。

要使用简短风格，必须保证 php.ini 文件中的 short_open_tag＝On（默认是开启的），本书中的 PHP 代码都采用这种风格。

3）脚本风格

这种风格将 PHP 代码写在<script>标记对中，例如：

```
<h1><script language='php'>echo '现在是'.date("m月 d日");</script></h1>
```

4）ASP 风格

这种标记风格将 PHP 代码写在“<%”和“%>”中，我们不推荐使用，并且默认是没有启用这种风格的，因为 php.ini 文件中默认是：asp_tags＝Off。

2. PHP 代码的注释

注释即代码的解释和说明，程序执行时，注释会被 PHP 解析器忽略，因此浏览器端看不到 PHP 代码的注释。PHP 支持 3 种风格的注释。

1）单行注释（//或#）

```
<? echo 'PHP动态网页';          //输出字符串
                               #单行注释用#号也可以  ? >
```

需要注意的是，单行注释的内容中不能含有“? >”，否则解释器会认为 PHP 的脚本到此结束了，而去执行“? >”后面的代码。导致多出一个“? >”而出错，例如：

```
<h1><? echo '这样会出错的';          //不会看到? >会看到
? ></h1>
```

2）多行注释（/ * … * /）

如果要添加大段的注释，则使用多行注释更方便，但多行注释符不允许嵌套使用。如：

```
<h1><?   echo '这样不会出错';                /* 多行注释的内容
不会被输出? >    * /   ? ></h1>
```

7.1.2 简单 PHP 程序示例

利用 PHP 程序可以输出字符串、HTML 代码或 JavaScript 代码，下面是几个例子。

1. 输出当前日期时间（7-1.php）

下面的程序以 h1 标题的形式输出当前日期和时间，代码如下：

```
<h1>
    <? echo '现在是'.date("Y年 m月 d日 H:i:s");? >
</h1>
```

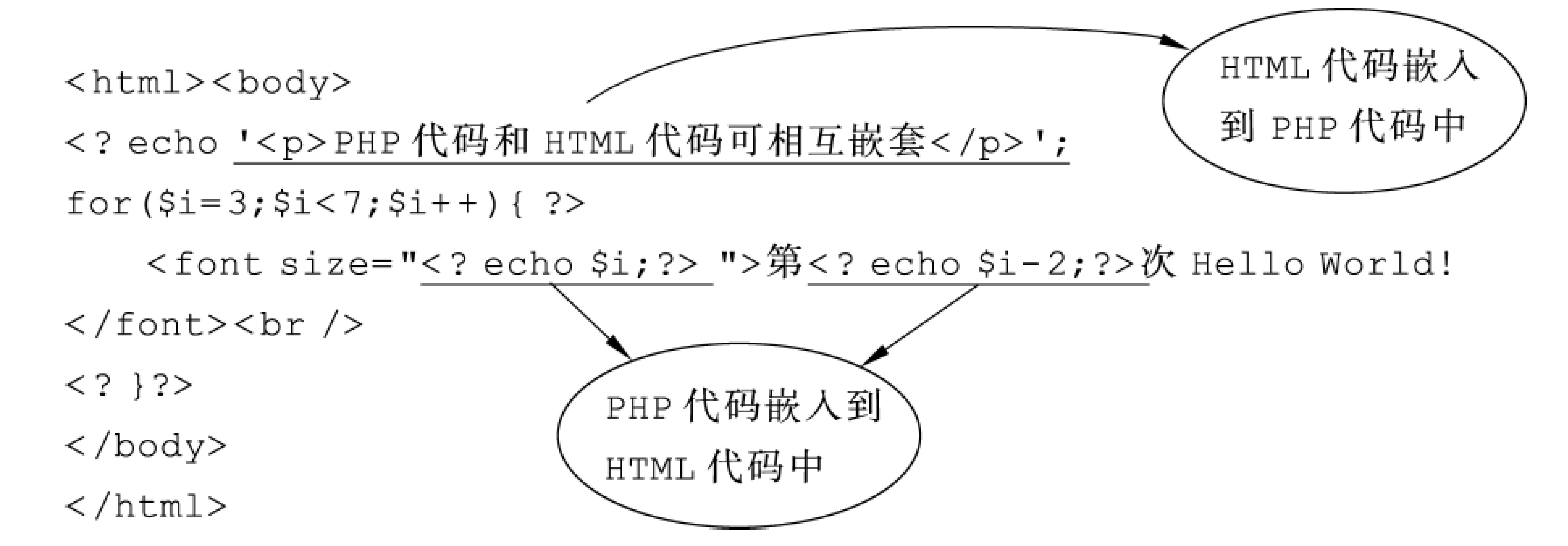
在该程序中，<h1>和</h1>是 HTML 代码，<? … ?>是 PHP 代码。其中，echo 是 PHP 的输出函数，单引号括起来的表示这是一个字符串常量，“.”是字符串连接符，

date()是时间日期函数，可以按指定的格式获取当前日期和时间。运行程序会在网页上以一级标题的形式显示：

现在是 2014 年 03 月 18 日 16:20:55

2. 输出不同大小的字体（7-2. php）

下面的程序使用 PHP 的循环语句重复输出<font>标记，其运行效果如图 7-1 所示。



```
<html><body>
<? echo '<p>PHP 代码和 HTML 代码可相互嵌套</p>';
for($i=3;$i<7;$i++){ ?>
    <font size="<? echo $i;?> ">第<? echo $i-2;?>次 Hello World!
</font><br />
<? }?>
</body>
</html>
```


在该程序中，使用 for 循环语句循环输出 HTML 代码“…
”。从结构上看，这条 HTML 代码被 PHP 代码包含。$i 是程序中定义的一个变量，PHP 规定所有变量名必须以“$”开头。可以看出，PHP 代码可以位于 HTML 代码的任意位置。如标记外：<? for($i=3;$i<7;$i++){?>、<? }?>，标记内：<? echo $i-2;?>，甚至是标记的属性内：<? echo $i;?>。从结构上看，可以是 HTML 代码中包含 PHP 代码，也可以是 PHP 代码中包含 HTML 代码。实际上，PHP 代码还可与 CSS 或 JavaScript 等浏览器端代码互相嵌入，因为 PHP 解析器只对“<?”和“?>”之间的代码进行处理。



图 7-1　7-2. php 或 7-3. php 的运行结果

注意：PHP 代码的定界符“<?”和“?>”不能够嵌套。如果遇到 HTML 代码（如<font…），就必须立即用“?>”把前面的 PHP 代码结束，即使这段代码并不完整（但其中每行语句必须是完整的）。

3. 用 PHP 程序输出 HTML 代码，实现 7-2. php 的功能（7-3. php）

在 7-2. php 中，由于 PHP 代码和 HTML 代码频繁地交替出现，以致经常需要使用定界符结束和开始一段 PHP 代码，而如果把 HTML 代码当成字符串通过 PHP 程序来输出，则可避免该问题。代码如下，运行效果如图 7-1 所示。

```
<html><body>
    <p>PHP 代码和 HTML 代码可相互嵌套</p>
```

```
    <? for($i=3;$i<7;$i++){
        echo '<font size='.$i .'>第' . ($i-2) .'次 Hello World!</font><br />';
    }? >
</body></html>
```

提示：使用 PHP 程序输出 HTML 代码是一项常用技巧。总的原则是：如果 PHP 代码之间的 HTML 代码很短，则使用 PHP 程序输出这些 HTML 代码更合适；而如果 PHP 代码之间的 HTML 代码很长，则还是作为外部 HTML 代码合适些。这样会使程序的可读性改善，并减少编写时出错的几率。

4. 输出 JavaScript 代码并传递变量值给 JavaScript 或表单(7-4.php)

下面的程序定义了两个变量 $ str1 和 $ str2，并将字符串赋值给这两个变量(PHP 中没有变量声明语句，变量不需要声明就可赋值使用)。因为 JavaScript 代码也是客户端代码，可以使用 PHP 将 JavaScript 代码作为字符串输出。如果在输出的 JavaScript 代码或表单代码中嵌入了 PHP 变量，就可以把这些服务器端变量值传递到客户端。

```
<?  $str1="Hello";                    //在弹出框中显示
    $str2="start PHP";                //在文本框中显示
    echo "<script>";
    echo "alert('".$str1."');";       //在 JavaScript 中使用$str1 变量
    echo "</script>";      ?>
<input type="text" name="tx" size=20 value="<? echo $str1; ?>">
<input type="button" value="单击" onclick="tx.value='<? echo $str2; ?>'">
```

运行该程序，会在弹出警告框和文本框中显示“Hello”，当单击按钮后，文本框中的内容会变为“start PHP”。

5. 编写 PHP 程序的注意事项

(1) PHP 是一种区分大小写的语言，表现在：

① PHP 中的变量和常量名是区分大小写的；

② 但 PHP 中的类名和方法名，以及一些关键字(如 echo、for)都是不区分大小写的。在书写时，建议除了常量名以外的其他名称都小写。

(2) PHP 代码中的字符均为半角(英文状态下)字符，中文或全角字符只能出现在字符串常量中。

(3) 在“<?”和“? >”内必须是一行或多行完整的语句，如“<? for($i=3; $i<7; $i++)? >”不能写成“<? for($i=3;? > <? $i<7; $i++)? >”。

(4) 在 PHP 中，每条语句以“;”号结束，PHP 解析器只要看到“;”号就认为一条语句结束了。因此，可以将多条 PHP 语句写在一行内，也可以将一条语句写成多行。

7.2 常量、变量和运算符

7.2.1 常量和变量

1. 常量

在程序运行中,其值不能改变的量,称为常量,常量通常直接书写,如 10、－3.6、"hello"都是常量,除此之外,还可以用一个标识符代表一个常量,称为符号常量。在 PHP 中使用 define()函数来定义符号常量,符号常量一旦定义就不能再修改其值。另外,使用 defined()函数可以判断一个符号常量是否已被定义。例如:

```
<?    define("PI","3.1416");             //定义符号常量 PI,并且区分大小写
define("SITE","网页设计学习网",true);      //定义符号常量 SITE,不区分大小写
echo (defined("PI"));                      //如果已被定义则返回“1”
?>
```

在 PHP 中,还预定义了一些符号常量,如表 7-1 所示(注意__FILE__等常量左右两边是双下划线),这些符号常量可直接使用,如“echo __FILE__;”。

表 7-1 PHP 预定义的符号常量

常　　量	功　　能
__FILE__	存储当前脚本的物理路径及文件名称
__LINE__	存储该常量所在的行号
__FUNCTION__	存储该常量所在的函数名称
PHP_VERSION	存储当前 PHP 的版本号
PHP_OS	存储当前服务器的操作系统名

2. 变量

变量是指程序运行过程中其值可以变化的量,变量包括变量名、变量值和变量的数据类型三要素。PHP 的变量是一种弱类型变量,即 PHP 变量无特定数据类型,不需要事先声明,并可以通过赋值将其初始化为任何数据类型,也可以通过赋值随意改变变量的数据类型。下面是一些变量声明(定义)和赋值的例子:

```
<?$str1="PHP 变量 1";          //该变量为字符串变量
$num=10+2*9;                   //该变量为数值型变量
$_date="2013-9-8";             //该变量为字符串变量,PHP 无日期型数据类型
$bol=true;                     //该变量为布尔型变量
$num='赋值字符串';             //通过赋值改变变量的数据类型
$str1=$num+$_date;             //执行相加运算会使$str1 转换为数值型,值为 2013
var_dump($num,$_date,$bol);    //var_dump 函数可输出变量的类型
?>
```

说明：

(1) PHP 变量必须以“$”开头，区分大小写；

(2) 变量在使用前不需要声明，PHP 中也没有声明变量的语句；

(3) 变量名不能以数字或其他字符开头，其他字符包括@、# 等。例如，$ xm、$ _id、$ sfzh 都是合法的变量名，而 $ -id、$ 57zhao、$ zh fen 都是非法的变量名；

(4) 变量名长度应小于 255 个字符，不能使用系统关键字作为变量名。

7.2.2 变量的作用域和生存期

1. 变量的作用域

变量的作用域是指该变量在程序中可以被使用的范围。对于 PHP 变量来说，如果变量是定义在函数内部的，则只有这个函数内的代码才可以使用该变量，这样的变量称为“局部变量”。如果变量是定义在所有函数外的变量，则其作用域是整个 PHP 文件，减去用户自定义的函数内部(注意这和 ASP VBScript 语言是不同的)，称为“全局变量”。例如：

```
<?  $a="全局变量<br>";         //该变量为全局变量
function fun(){
    echo $a;                 //调用函数也不会输出“全局变量”
    $a="局部变量、";           //该变量为局部变量
    echo $a;
}
fun();                       //输出“局部变量”
echo $a;                     //输出“全局变量”    ?>
```

输出结果为“局部变量、全局变量”。可见函数内不能访问函数外定义的变量。

如果一定要在函数内部引用外部定义的全局变量，或者在函数外部引用函数内部定义的局部变量。可以使用 global 关键字。示例代码(global.php)如下：

```
--------------------------global.php--------------------------
<?  $a="全局变量、";
function fun(){
    global $a;               //为了引用函数外定义的变量$a
    echo $a;
    $a="局部变量、";           //将修改$a的值,添加 static试试
    echo $a;                 //输出“局部变量”
    }
fun();                       //调用函数将输出“全局变量、局部变量、”
echo $a;                     //输出“局部变量、”
?>
```

输出结果为“全局变量、局部变量、局部变量、”。

提示：

(1) global的作用并不是将变量的作用域设置为全局，而是起传递参数的作用。在函数外部声明的变量，如果想在函数内部使用，就在函数内用global来声明该变量。

(2) 不能在用global声明变量的同时给变量赋值。例如，“global ＄a＝"全局"”是错误的。

(3) global只能写在自定义函数内部，写在函数外部没有任何用途。

(4) 对于global变量，应该用完之后就用unset()销毁，因为它很占用资源。

另外，使用＄GLOBALS[]全局数组也能实现在函数内部引用外部变量，例如：

```
<?  $a="全局变量、";
function fun(){
    echo $GLOBALS['a'];
    $a="局部变量、";
    echo $a;                    //输出“局部变量”
    }
fun();                          //调用函数将输出“全局变量、局部变量”
echo $a;                        //输出“全局变量”
?>
```

则输出结果为“全局变量、局部变量、全局变量”。可见，＄GLOBALS[]和global是有区别的，它只能在函数内部引用外部变量，但不能在函数外部引用函数内部定义的局部变量。

2. 变量的生存期

变量的生存期表示该变量在什么时间范围内存在。全局变量的生存期从它被定义那一刻起到整个脚本代码执行结束为止；局部变量的生存期从它被定义开始到该函数运行结束为止。

可见，一般的局部变量在函数调用结束后，其存储的值会被自动清除，所占存储空间也会被释放。为了能在函数调用结束后仍保留局部变量的值，可使用静态变量，这样当再次调用函数时，又可以继续使用上次调用结束后的值。静态变量使用static关键字定义。例如：

```
<?  function Test() {
        static $w=0;        //声明静态变量$w
        echo $w;
        $w++;       }
Test();Test();Test();Test();Test();     ?>
```

程序的输出结果为“01234”。而如果去掉程序中的“static”，则运行结果为“00000”。

提示：

(1) 静态变量仅在局部函数域中存在，函数外部不能引用函数内部的静态变量。例如将global.php中的“＄a＝"局部变量";”改为“static ＄a＝"局部变量";”，则最后一条

语句将输出“全局变量”。

(2) 对静态变量赋值时不能将表达式赋给静态变量。如“static $int=1+2; static $int=sqrt(9);”都是错误的。

表 7-2 对 3 种类型的变量进行了总结。

表 7-2 变量根据作用域和生存期分类

类 型	说 明
全局变量	定义在所有函数外的变量,其作用域是整个 PHP 文件,减去用户自定义的函数内部
局部变量	定义在函数内部的变量,只有这个函数内的代码才可以使用该变量
静态变量	局部变量的一种,能够在函数调用结束后仍保留变量的值

7.2.3 可变变量和引用赋值

1. 可变变量

可变变量是一种特殊的变量,这种变量的名称不是预先定义的,而是动态地设置和使用的。可变变量一般是使用一个变量的值作为另一个变量的名称,所以可变变量又称为变量的变量。可变变量直观上看就是在变量名前加一个“$”。例如:

```
<?  $a='b';                    //定义变量$a
    $b='一个变量<br>';          //定义变量$b
    echo $$a;                  //$$a就是一个可变变量,相当于$b
    $b='变化后';
    echo $$a;                  //通过可变变量输出变量$b的值
    $a='c';
    echo $$a;                  //相当于输出变量$c的值
    ?>
```

输出结果是“一个变量
变化后”。由于没有给 $c 赋值,第三条 echo 语句不会输出任何内容。

2. 引用赋值

从 PHP 4.0 开始,提供了“引用赋值”功能。即新变量引用原始变量的地址,修改新变量的值将影响原始变量,反之亦然。引用赋值使得不同的变量名可以访问同一个变量内容。使用引用赋值的方法是:在将要赋值的原始变量前加一个“&”符号。例如:

```
<?  $b=10;
$a="hello";                    //$a赋值为 hello
$b=&$a;                        //变量$b引用$a的地址
echo $a;                       //输出结果为 hello
$b="world";                    //修改$b的值,$a的值将一起变化
echo $a;                       //输出结果为 world
```

```
$a="cup";                      //修改$a的值,$b的值将一起变化
echo $b;                       //输出结果为 cup
?>
```

引用赋值的原理如图 7-2 所示。引用赋值后，两个变量指向同一个地址单元，改变任意一个变量的值（即地址中的内容），另一个变量值也会随之改变。

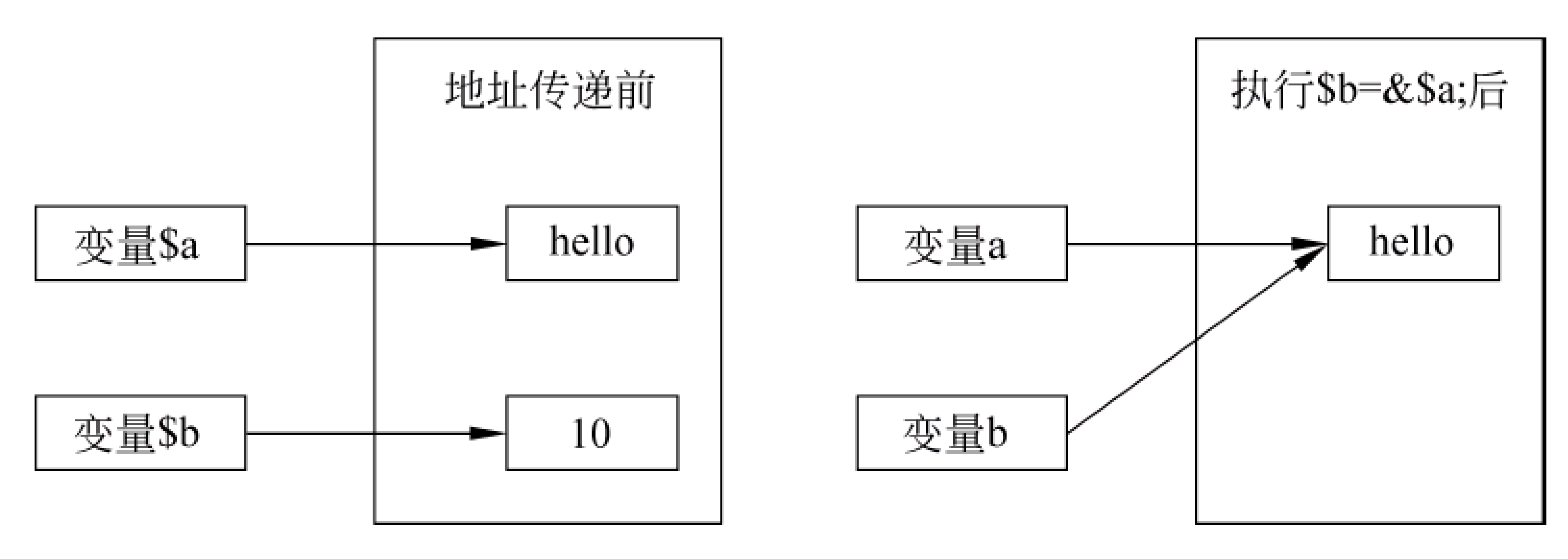


图 7-2 引用赋值——变量地址传递示意图

注意：只有已经命名过的变量才可以引用赋值，例如下面的用法是错误的：$bar=&(25*5)。

7.2.4 运算符和表达式

PHP 的运算符包括算术运算符、比较运算符、逻辑运算符、赋值运算符、连接运算符等。而表达式就是由常量、变量和运算符组成的，符合语法要求的式子。PHP 主要有 5 种表达式，即数学表达式（如 3+5*7）、字符串表达式（如"abc"."gh"）、赋值表达式（如 $a+=$b）、关系表达式（如 i==5）和逻辑表达式（如 $a||$b&&$c）。

1. 算术运算符

算术运算符有：加（+）、减（-）、乘（*）、除（/）、取余（%）等。算术运算符的运算结果是一个算术值。例如：$a=7/2+4*5+1，结果是 24.5。$b=-7%3，结果是-1（对于取余运算符来说，如果被除数是负数，那么取得的结果也是负数）。如果对 10 求余可得到一个数个位上的数字。

如果算术运算符的左右两边有一操作数或两个操作数都不是数值型时，那么会将操作数先转换成数值型，再执行算术运算。

例如：$a=10+'20'，结果为 30。$a='10'+'20'，结果为 30。$a='10'+'2.2ab8'，结果为 12.2。$a='10'+'ab2.2'，结果为 10。$a='10'+true，结果为 11。

字符串转换为数值型的原则是：从字符串开头取出整数或浮点数，如果开头不是数字的话，就是 0。布尔型的 true 会转换成数值 1，false 转成数值 0。

2. 连接运算符

PHP 中连接运算符只有一个，即“.”，它用于将两个字符串连接起来，组成一个新字符串。如果连接运算符左右两边任一操作数或两个操作数都不是字符串类型，那么会将操作数先转换成字符串，再执行连接操作。例如：

```
$a= 'PHP'. 5;          //$a的值为 PHP5,注意数字和"."之间要用空格隔开
$b= 'PHP'.'5';         //$b的值为 PHP5
$c= "PHP".True;        //$c的值为 PHP1
$d= 5 . 'PHP';         //$d的值为 5PHP
```

提示：如果“.”的左右有数字，注意将“.”和数字用空格隔开。

可见“.”是强制连接运算符，不管左右两边是什么数据类型，都会执行连接运算。

3. 赋值运算符

最基本的赋值运算符是“=”，它用于对变量赋值，因此它的左边只能是变量，而不能是表达式。例如：$a=3+5，$b=$c=9 都是合法的。此外，PHP 还支持像 C 语言那样的赋值运算符与其他运算符的缩写形式，如“+=”、“.=”、“&=”、“|=”等。

如 $a+=3 等价于 $a=$a+3，$a.=3 等价于 $a=$a. 3。

4. 比较运算符

比较运算符会比较其左右两边的操作数，如果比较结果为真，则返回 true，否则返回 false。PHP 中的比较运算符有：是否相等(==)、大于(>)、小于(<)、大于等于(>=)小于等于(<=)、不等于(!=或<>)、恒等于(===)、非恒等于(!==)。

其中恒等于(===)表示数值相等并且数据类型也相同，非恒等于(!==)表示数值不相等或者数据类型不相等。

例如，若 $a=6，$b=3，则 $a<$b 返回 false，$a>$b 返回 true，$a<>$b 返回 true。$c="PHP"<"php"返回 true。$c="5"==5 返回 true，$c="5"===5 返回 false，$c=1==true 返回 true，$c=1!==true 返回 true。

5. 逻辑运算符

逻辑运算符用来组合逻辑运算的结果，比如对两个布尔值或两个比较表达式进行逻辑运算，再返回一个布尔值(true 或 false)。PHP 中的逻辑运算符有逻辑非(!)、逻辑与(&& 或 and)、逻辑或(|| 或 or)、逻辑异或(xor)。

例如：!5<3&& 'b'=="b"返回 true，!(5>3&& 'b'==="b")返回 false。

逻辑与(&& 和 and)，逻辑或(|| 和 or)，虽然含义相同，但它们的优先级却不同。“&&”的优先级比“and”高，“||”的优先级比“or”高。例如：$c=(1 or 2 and 0)，会返回 true。因为 or 和 and 优先级相同，则按自右至左的执行顺序，先执行 2 and 0。

而 $c=(1||2 and 0)，会先执行 1||2，再执行 true and 0，最终返回 false。

又如 $c=false or 1，返回 false，$c=false||1，返回 true。因为“=”的优先级比“or”高，但是比“||”低。

6. 加 1/减 1 运算符

加 1/减 1 运算符与 C 语言中的加 1/减 1 运算符相同，包括前加(++$a)、后加($a++)、前减(--$a)、后减($a--)4 种形式。

前加操作是先加 1，再赋值，后加操作是先赋值，再加 1。例如，$a=6；$b=++$a，则执行完后，$a=7，$b=7。$a=6；$b=$a++，执行完后，$a=7，$b=6。

前减操作和后减操作的规则与此相同。

7. 条件运算符

条件运算符是一个三元运算符，其语法如下：

```
条件表达式 ?表达式 1:表达式 2
```

如果条件表达式的结果为 true，则返回表达式 1 的值，否则返回表达式 2 的值。

例如，下面的表达式会得到“Yes”。

```
$c=10>2 ? "Yes": "No"
```

在分页程序中，常通过条件运算符判断要显示的分页页面，如果获取的分页变量 page 的值存在，则显示该分页，如果获取不到 page 变量值，则显示第 1 页，代码如下：

```
$page=(isset($_GET['page']))?$_GET['page']:"1";
```

8. 执行运算符

执行运算符，即反引号(` `)(键盘上的反引号键在数字 1 键的左边)。可用来执行 Shell 命令。在 PHP 脚本中，将外部程序的命令行放入反引号中，并使用 echo()或 print()函数将其显示，PHP 将会在到达该行代码时启动这个外部程序，并将其输出信息返回，其作用效果与 shell_exec()函数相同。例如：

```
<? $output=`dir`;
echo $output;                    //输出当前目录下的内容
echo shell_exec('dir ');         //输出当前目录下的内容,结果同上?>
```

提示：IIS 出于安全性考虑，禁止使用执行运算符，执行运算符只能在 Apache 中使用。

7.3　数据类型及类型转换

数据类型是一个值的集合以及定义在这个值集上的一组操作。数据类型的使用往往和变量的定义联系在一起。虽然 PHP 定义变量时不需要指定数据类型，但它会根据对变量所赋的值自动确定变量的数据类型。确定了变量的数据类型就确定了变量的存储方式(占多少字节)和操作方法。PHP 具有的数据类型如表 7-3 所示。

表 7-3　PHP 中的数据类型

数据类型	具体描述
整型(integer)	即整数，占 4 个字节(32 位)，取值范围从 −2147483648 到 2147483647 之间，可以采用十进制、八进制(0 作前缀)、十六进制(0x 作前缀)表示

续表

数 据 类 型	具 体 描 述
浮点型(float)	即实数(包含小数的数),如 1.0、3.14
布尔型(boolean)	只有 true(逻辑真)和 false(逻辑假)两种取值
字符串(string)	是一个字符的序列。组织字符串的字符可以是字母、数字或者符号
数组(array)	由一组相同数据类型的元素组成的数据结构,每个元素都有唯一的编号
对象(object)	是面向对象语言中的一种复合数据类型,对象就是类的一个实例
NULL	空类型,只有一个值 NULL。如果变量未被赋值,或被 unset()函数处理后的变量,其值就是 NULL
资源(resource)	资源是 PHP 特有的一种特殊数据类型,用于表示一个 PHP 的外部资源,比如一个数据库的访问操作,或者打开保存文件操作。PHP 提供了一些特定的函数,用于建立和使用资源
伪类型	只用于函数定义中,表示一个参数可接受多种类型的数据,还可以接受别的函数作为回调函数使用

7.3.1 字符串数据类型

任何由字母、数字、文字、符号组成的 0 到多个字符的序列都叫作字符串。在 Web 程序中,经常需要对字符串进行操作。如截取标题、连接字符串常量和变量等。PHP 规定字符串的前后必须加上单引号(')或双引号("),例如,"这是一个字符串"、'另一个字符串'、'5'、'ab'、"(空字符串)都是合法的字符串。但单双引号不能混用,如'day"是非法的。

如果字符串中出现单引号(')或双引号("),则需要使用转义字符(\'或\")来输出,例如:

```
echo 'I\'m a boy';          //输出结果为 I'm a boy
```

1. 单引号字符串和双引号字符串

单引号表示包含的是纯粹的字符串;而双引号中可以包含字符串和变量名。双引号中如果包含变量名则会被当成变量,会自动被替换成变量值,单引号中的变量名则不会被当成变量,而是把变量名当成普通字符输出。示例代码如下,运行效果如图 7-3 所示。

```
<?$a='tang';
$b=10;
echo '你好$a';              //使用单引号输出$a
echo '<br>';
echo "你好$a";              //使用双引号输出变量
echo "你是第$b次光临";      //使用双引号输出变量?>
```



图 7-3 单引号字符串与双引号字符串

可见,在双引号字符串中,$a 和 $b 被解析成了变量

$ a 和 $ b 的值。因此建议：如果要书写纯字符串，建议用单引号字符串；如果要对字符串和变量进行连接操作，可以使用双引号字符串，以简化写法，例如以下两条语句是等价的：

```
echo "你是第$b 次光临";                //注意$b后面要有个空格
echo '你是第 '.$b .' 次光临';          //与上一条语句完全等价
```

注意：在双引号字符串中，如果变量名后有其他字符的话，要在变量名后加空格，否则 PHP 解析器会认为后面的字符也是变量名的一部分。例如：

```
$sport= 'basket';
$hobby= "I like play$sportball.";      //包含变量的错误方法
echo $hobby;
```

则 PHP 解析器认为双引号中的变量是 $ sportball，而 $ sportball 未定义，视为值为空。因此会输出“I like play .”。为解决这个问题，可以加空格，或用大括号将变量名包含起来。

```
$hobby= "I like play {$sport}ball.";   //包含变量的正确方法
$hobby= "I like play$sport ball.";     //另一种写法，$sport 后有个空格
```

双引号字符串比单引号字符串支持更多的转义字符，双引号字符串支持的转义字符如表 7-4 所示。

表 7-4　双引号字符串支持的转义字符

转义字符	含义	转义字符	含义	转义字符	含义
\n	换行	\t	跳格 Tab	\\	反斜杠\
\r	回车	\"	双引号	\$	显示 $ 符号

例如，要在双引号字符串中输出 $ 符号、反斜杠和换行符。代码如下：

```
echo "变量\$a= '\\t'\n";        //输出结果为：变量$a= '\t'(换行)
```

但是换行符会被浏览器当成空格忽略，只有在网页源代码中才能看到换行符的效果。

2. 界定符表示字符串

除了使用单引号或双引号表示字符串外，还可使用界定符表示字符串或变量，例如：

```
<?  $i= '显示该行内容';
    echo <<<STD
    双引号""可直接输出,\$i 同样可以被输出出来。<br>
    \$i 的内容为:$i
    STD;
?>
```

输出结果为：

```
双引号""可直接输出,$i同样可以被输出出来。
$i的内容为:显示该行内容
```

说明:

(1) 程序中的"STD"是自定义的界定符,也可以使用任何其他标识符,只要首尾界定符相同即可。

(2) 开始界定符前面必须有3个左尖括号"<<<",后面不能有任何空格。结束界定符必须单独另起一行,前后不能有空格或任何其他字符(包括注释符),否则都会引起语法错误。

(3) 界定符和双引号唯一的区别是界定符中的双引号不需要转义就能显示,因此,如果需要处理大量的内容,同时又不希望频繁使用各种转义字符,则使用界定符更合适。

3. 获取字符串中的字符

在PHP中,可以通过给字符串变量加下标的方式获取字符串中的字符。语法为:

```
字符串变量[index]
```

其中,index指定字符的位置,0表示第1个字符,1表示第2个字符,以此类推。但该方法不能用来获取中文字符,否则会出现乱码,因为一个中文占两个字符。例如:

```
<? $i='Tom & Mary';
   echo $i[1] .$i[4];          //输出结果为o&,因为空格也算一个字符
?>
```

4. 获取字符串的长度

使用strlen()函数可获取字符串的长度,该函数的参数是一个字符串,例如:

```
<? echo strlen('喜欢 PHP!');  ?>
```

输出的结果是8,这是因为每个中文字符占2个字节,加上后面4个英文字符总共占8个字节。如果要计算中文字符串的长度,可以使用mb_strlen()函数,例如:

```
<? echo mb_strlen('喜欢 PHP!',"gb2312");    ?>
```

则返回值为6,将网页字符编码设置为GBK或gb2312即可获得正确的中文字符串长度。

如果要判断字符串的长度是否要小于某个值,有以下两种方法:

```
if(strlen($pwd)<6) echo "密码太短";
if(!isset($pwd{6})) echo "密码太短";          //如果第7个字符不存在
```

而第二种方法不需要使用函数,效率更高。

7.3.2 数据类型的转换

PHP中数据类型的转换有以下两种情况。

1. 自动类型转换

（1）如果对变量重新赋了不同数据类型的值，则变量的数据类型会自动转换，例如：

```
$a="Hello";          $a=12;
```

则变量＄a 的数据类型就会由字符串型转换成整型。

（2）如果不同数据类型的变量进行运算操作，则将选用占字节最多的一个运算数的数据类型作为运算结果的数据类型。例如：

```
$a=1+3.14;
$b=2+"2.0";
$c=3+"php";
var_dump($a,$b,$c);          //输出 float(4.14) float(4) int(3)
```

则＄a 的数据类型为浮点型。在第 2 个赋值表达式中，首先将字符串数据"2.0"转换成浮点型数据 2.0，然后进行加法运算，赋值后＄b 的数据类型为浮点型。在第 3 个表达式中，首先将字符串数据转换成整型数据 0，然后进行加法运算，赋值后＄c 的数据类型为整型。

2. 强制数据类型转换

利用强制类型转换可以将数据类型转换为指定的数据类型。其语法如下：

```
(类型名) 变量或表达式
```

其中类型名包括 int、bool、float、double、real、string、array、object，类型名两边的括号一定不能省略。例如：

```
$a="2.0";              $b=(int)$a;
$c=(array)$a;          print_r($c);
```

则＄b 将转换成整型。＄c 将转换为数组类型 Array([0]=>2.0)。

虽然强制数据类型转换使用起来很方便，但也存在一些问题，比如字符串型转换成整型该如何转换，整型转换成布尔型该如何转换，这些都需要一些明确的规定，PHP 为此提供了相关的转换规定，如表 7-5 所示。

表 7-5　PHP 类型转换的规定

源类型	目的类型	转换规则
float	integer	保留整数部分，小数部分无条件舍去
Boolean	integer 或 float	false 转换成 0，true 转换成 1
Boolean	string	false 转换成空字符串""，true 转换成字符串"1"
string	integer	从字符串开头取出整数，开头没有的话，就是 0。例如字符串"3M"、"8.6uc"、"x5"会转换成整数 3、8、0
string	float	从字符串开头取出浮点数，开头没有的话，就是 0.0。例如字符串"3M"、"8.6uc"、"x5"会转换成整数 3.0、8.6、0.0

续表

源类型	目的类型	转换规则
string	boolean	空字符串""或字符串"0"转换成 false，其他都转换成 true，因此字符串"false"也会转换成 true
integer float	boolean	0 转换成 false，非 0 的数都转换成 true
integer float	string	将所有数字转换成字符串，如 12 转换成"12"，3.14 转换成"3.14"
integer float boolean string	array	创建一个新的数组，第一个元素就是该整数、浮点数、布尔值或字符串
array	string	字符串"Array"
object	boolean	没有成员的对象转换成 false，否则会转换成 true

提示： 如果使用 echo 函数输出布尔值 echo true，则会输出字符串"1"；若输出 echo false，则会输出空字符串。因为任何数据类型输出时都将被转换成字符串。

7.4 PHP 的语句

PHP 的语句可分为顺序执行语句、条件控制语句、循环控制语句以及包含语句等。

7.4.1 条件控制语句

在 PHP 中，有 if 语句和 switch 两种条件语句。if 语句又可分为单分支选择 if 语句、双分支选择 if 语句和多分支选择 if 语句三种。

1. 单分支选择 if 语句

一般形式为：

```
if(条件表达式) {
    语句块  }
```

它表示当条件表达式成立时(值为 true)，执行“语句块”。例如：

```
if($sex==1) echo "尊敬的先生";
```

如果语句块中包含多条语句，则要使用{}将这些语句包含起来，使它们构成一条复合语句。例如：

```
if($a>$b)     {
    $temp=$a;
    $a=$b;
    $b=$temp;}
```

2. 双分支选择 if…else 语句

一般形式为：

```
if(条件表达式)
    {  语句块 1  }
else
    {  语句块 2  }
```

表示当条件表达式值为 true 时,执行“语句块 1”,否则执行“语句块 2”。例如:

```
if($a)$a=0; else $a=1;
```

该语句被称为“开关语句”。即如果 $a 的值为 true 或非 0,则让 $a 的值为 0;否则让 $a 的值为 1,因此每执行一次都会使 $a 的值在 0 和 1 之间转换。if($a)是 if($a== true)的简写形式。

3. 多分支选择 if…elseif…else 语句

一般形式为:

```
if(表达式 1)      语句块 1
elseif(表达式 2)      语句块 2
elseif(表达式 3)      语句块 3
…
else 语句块 n
```

它会首先判断表达式 1 是否成立,如果成立,则执行语句块 1,执行完后,直接退出该选择结构,不再判断后面的表达式是否成立。如果表达式 1 不成立,则再依次判断表达式 2 到表达式 n 是否成立;如果成立,则执行对应的语句块 i;如果所有表达式都不成立,则执行 else 后的语句块 n。例如,要找出 3 个数中的最大数,程序如下:

```
if($a<$b)$max=$b;
elseif($a<$c)$max=$c;
else $max=$a;
```

说明:

(1) if 语句还可以嵌套使用,也就是说,“语句块”中还可以存在 if 语句。

(2) if(条件表达式)后一般没有“;”号,如果有“;”,表示 if 语句的语句块为空语句。

(3) 语句块如果是一条语句则后面一定要有“;”,如果语句块是由{}包含的复合语句,则{}后不要有“;”。

4. switch/case 语句

switch 语句是多分支选择 if 语句的另一种形式,两者可互相转换。在要判断的条件有很多种可能的情况下,使用 switch 语句将使多分支选择结构更加清晰。一般形式为:

```
switch(变量或算术表达式)  {
    case(常量 1):    语句块 1
    case(常量 2):    语句块 2
    …
```

```
    case(常量 n):    语句块 n
    default:语句块 n+1
}
```

下面的程序根据时间显示不同的欢迎信息(7-5.php)。

```
<? $a=date(G);        //获取当前时间的小时数
$a=floor($a/3);       //将小时数除以 3并取底
switch ($a)  {
    case 2:echo "早上好";break;
    case 3:echo "上午好";break;
    case 4:case 5: echo "下午好";break;
    case 6:echo "晚上好";break;
    default: echo "该睡觉了";break;
}   ?>
```

说明:

(1) case 语句后不能接表示范围的条件表达式,只能接常量。

(2) 每个 case 中的常量必须不同,如果相同,则满足条件时只会执行最前面 case 语句中的内容。

(3) 多个 case 可共用一组语句,此时必须写成"case 4:case 5:"的形式,不能写成"case 4,5:"。

(4) 每个 case 后一般都要有一条 break 语句,这样执行完该 case 语句后就会跳出分支结构,否则,执行完该 case 语句后还会依次执行下面的 case 语句,直到遇到 break 或执行完。

(5) 各个 case 和 default 语句的出现顺序可随意变动。

7.4.2 循环控制语句

循环结构通常用于重复执行一组语句,直到满足循环结束条件时才停止。在 PHP 中,主要有 4 种循环语句,即 for 循环、foreach 循环、while 循环和 do…while 循环。

1. for 循环

for 循环语句是不断地执行循环体中语句,直到相应条件不满足,并且在每次循环后处理计数器。for 语句的一般形式为:

```
for(初始表达式;循环条件表达式;计数器表达式)
    {循环体语句块 }
```

其执行过程为:

(1) 执行初始表达式(通常是给循环变量赋初值);

(2) 判断循环条件表达式是否成立,若成立,则执行循环体,否则跳出循环;

(3) 执行一遍循环体语句块;

(4) 执行计数器表达式(通常是给循环变量计数);

(5) 转到第(2)步判断是否继续循环。

循环可以嵌套,例如要用 for 循环画金字塔,有下面两种写法,运行效果如图 7-4 所示。

(1) 写法 1(7-6.php)

```
<div align="center">
<?
for($i=0;$i<5;$i++){
      for($j=0;$j<=$i;$j++)
        echo "*  ";
      echo "<br/>";
}
?></div>
```

(2) 写法 2(7-7.php)

```
<div align="center">
<?
for($i=0;$i<5;$i++){
  $a=$a ."*  ";
  echo $a ."<br/>";
}
?>
</div>
```

提示:在对矩阵进行操作时,通常需要双重循环嵌套。

2. foreach 循环

foreach 语句通常用来对数组或对象中的元素进行遍历操作,例如数组中的元素个数未知,则很适合使用 foreach 语句。其一般形式为:

```
foreach(数组名 as $ value)
     { 循环体语句块 }
```

或者

```
foreach(数组名 as $key=>$value)
     { 循环体语句块 }
```

foreach 语句遍历数组时首先指向数组中第 1 个元素。每次循环时,将当前数组元素值赋给 $ value,将当前数组索引值赋给 $ key,再让数组指针向后移动直到遍历结束。示例程序如下,运行效果如图 7-5 所示。

```
<?                          //7-8.php
$sports=array("网球","游泳","短跑","柔道");      //定义并初始化一个数组
echo "我校开展的运动项目有:<br />";
foreach($sports as $key=>$value)
     echo $key .":".$value . " ";          ?>
```



图 7-4　画金字塔程序



图 7-5　foreach 示例程序

3. while 循环

while 语句是前测试循环，即是否终止循环的条件判断是在执行循环体之前，因此循环体可能一次都不会执行。其一般形式为：

```
while(条件表达式) {
    循环体语句块 }
```

例如，要输出一个有 3 行的 html 表格，程序(7-9. php)如下，运行效果如图 7-6 所示。

```
<table border="1" width="300" align="center">
<?  $i=0;
while($i<3){
    echo "<tr><td>这是第$i 行</td></tr>";        //输出表格行
    $i++;
}  ?></table>
```



图 7-6 while 语句的应用

4. do…while 循环

do…while 语句是后测试循环，它将条件判断放在循环之后，这就保证了循环体中的语句块至少会被执行一次，在某些时候这是非常有用的。其一般形式为：

```
do {
    循环体语句块 }
while(条件表达式);          //注意 while (…)后有;号
```

例如，下面的程序会输出段落<p>元素一次：

```
<?  $i=0;
do{
    echo "<p>不满足循环条件,仍然会输出一次</p>";
    $i++;     }
while($i>1);  ?>
```

想一想：如果将 while($i>1)改成 while($i>0)，程序会循环多少次？

5. break 语句

break 语句用来提前终止循环，它可以出现在 while、do…while、for、foreach 或 switch 语句内部，用来跳出循环语句或 switch 语句。在用"穷举法"解题时，通常找到解后就用 break 终止循环。例如，要输出一个字符串，各元素之间用","号隔开，最后一个元素后没有","号，代码如下(7-10. php)：

```
<?  $sports=array("网球","游泳","短跑","柔道");
for($i=0;$i<4;$i++){
    echo $sports[$i];
```

```
        if($i==3) break;          //最后一个元素不输出",",换成 continue 试试
        echo ",";
    }  ?>
```

输出结果为:

网球,游泳,短跑,柔道

提示:在 PHP 中,break 后还可带参数 n,表示跳出 n 层循环,如"break 2;"会跳出 2 层循环,而其他语言的 break 语句一般不能带参数,只能跳出最近的一层循环。

6. continue 语句

continue 语句用来提前结束本次循环,即不再执行本次循环中 continue 语句后的语句,接着再执行下次循环,因此它不会提前终止循环。例如要用单个循环输出一个三行三列的表格,代码如下(7-11. php),运行结果如图 7-7 所示。

```
<table border="1" width="200" align="center"><tr>
<?  $i=0;
while($i<9){
    echo "<td>第$i 格</td>";        //输出表格的单元格
    $i++;
    if($i%3<>0||$i==9) continue;
    echo "</tr><tr>";
}  ?>
</tr></table>
```



图 7-7　continue 语句的应用

提示:如果正好是最后一次循环时用 continue 语句结束本次循环,那就相当于提前终止循环,这种情况下 continue 和 break 可互换。如 7-10. php 中的 if($i== 3) break 可换成 continue,因为 $i=3 时正好是最后一次循环。

7.4.3　文件包含语句

为了提高代码的重用性,可以将一段公用的代码保存为一个单独的文件,然后在其他需要这段代码的文件中,使用包含语句将文件引入。PHP 提供了 4 种包含语句。

1. include 语句

使用 include 语句可以在指定的位置包含一个文件,语法如下:

```
include(path/filename);              //括号可省略
```

当一个文件被包含时,编译器会将该文件的所有代码嵌入到 include 语句所在的位置。

例如,文件 7-12. php 和 7-13. php 位于同一目录下,在 7-12. php 中可以使用 include 语句将 7-13. php 包含进来。代码如下:

```
<?          //7-12.php
echo "我的名字是$name<br>";
include('7-13.php');          //也可嵌入 7-14.html
echo "我的名字是$name,我今年$age 岁。";
?>
```

```
<?          //7-13.php
$name='tangsix';
$age=33;          ?>
```

则编译器在执行 7-12.php 前，会把 7-13.php 的代码嵌入到 7-12.php 中，7-12.php 的运行结果如图 7-8 所示。可见 7-12.php 等价于：

```
<?     echo "我的名字是$name<br>";
$name='tangsix';          //嵌入的 7-13.php 的代码
$age=33;
echo "我的名字是$name,我今年$age 岁";
?>
```



图 7-8 7-12.php 的运行结果

include 语句也能包含 html 文件，这时 include 语句会自动使用"?>"将前面的 php 代码结束，用"<?"开始后面的 php 代码。例如将 7-12.php文件中 include('7-13.php')换成 include('7-14.html')。7-14.html 代码如下：

```
<center>&copy; 程序员实验室 版权所有</center>
```

则 7-12.php 等价于：

```
<?  echo "我的名字是$name<br>";     ?>
<center>&copy; 程序员实验室 版权所有</center>
<?  echo "我的名字是$name,我今年$age 岁";
?>
```

提示：如果被包含文件与包含文件不在同一目录中，则需要在被包含文件的文件名前加路径。有 3 个特殊的路径："../"代表上一级目录，"./"代表当前目录，"/"代表网站根目录(不是虚拟目录)。例如，"include('../file4.php');"将包含位于当前目录上一级目录中的 file4.php。

2. include_once 语句

include_once 语句与 include 语句相似，也用于包含文件。唯一的区别是：使用 include_once 包含文件时，如果该文件中的代码已被包含过，则不会再次包含。这样可以避免出现函数重定义、变量重新赋值等问题。

3. require 语句

require 语句也用于包含文件,但与 include 语句在错误处理上的方式不同。当包含文件失败时(如包含文件不存在),require 语句会出现致命错误,并终止程序的执行,而 include 语句只会抛出警告信息并继续执行程序。

4. require_once 语句

require_once 语句与 require 语句功能相似。但它在包含文件之前会先检查当前文件代码是否被包含过,如果该文件已经被包含了,则该文件会被忽略,不会被再次包含。

一般来说,由于 include 语句在出错时不会终止程序的执行,并显示警告信息,这容易产生安全问题,因此建议包含 PHP 程序文件时尽量使用 require 或 require_once 语句。

7.5 数组

数组是按一定顺序排列,具有某种数据类型的一组变量的集合。数组中的每个元素都是一个变量,它可以用数组名和唯一的索引(又称“下标”或“键名”)来标识。

7.5.1 数组的创建

PHP 中创建数组有两种方法:①使用 array()函数创建;②直接给数组元素赋值。

1. 使用 array()函数创建数组

PHP 的数组不需要定义,可以直接创建。创建数组一般使用 array()函数,假设要创建一个包含 4 个元素的一维数组,简单形式是:

```
$citys=array("长沙","衡阳","常德","湘潭");
```

则该数组的长度为 4。各个数组元素的索引值分别为:0、1、2、3,如 $citys[1]表示第 2 个数组元素。

如果要自行给每个元素的索引赋值,可以使用完整形式定义:

```
$citys=array('cs'=>'长沙','hy'=>'衡阳','cd'=>'常德','xt'=>'湘潭');
```

可见,完整形式增加了对数组元素索引的赋值,这时各个数组元素的索引值分别为:cs、hy、cd、xt,如 $citys[hy]表示第 2 个数组元素(注意此时不能再使用 $citys[1]访问该数组元素)。

2. 直接给数组元素赋值创建数组

也可以创建一个空数组,然后再给每个数组元素赋值。例如:

```
$citys=array();              //创建空数组,该语句可省略
$citys[1]="长沙";            $citys[3]="常德";     $citys[]="湘潭";
```

```
print_r($citys);                //打印数组
```

上述代码的输出结果为：

```
Array([1]=>长沙 [3]=>常德 [4]=>湘潭)
```

可见，如果给数组元素赋值时不写该元素的索引值，则该数组元素默认的索引值为数组中最大的索引值加1。如果数组中没有正整数形式的索引值，则默认的索引值为0。

实际上创建空数组的语句也可省略，那样就是通过直接给数组元素赋值创建数组了。

3. 创建数组注意事项

(1) 如果数组元素的索引是一个浮点数，则索引将被强制转换为整数，如$citys[3.5]将转换成$citys[3]。如果索引是布尔型数据，则将被强制转换成1或0，如果索引是一个整数字符串，则将被强制转换成整数，如$citys["3"]将转换成$citys[3]。

(2) 如果数组元素的索引是字符串，则最好要给索引加引号，如$citys=array('cs'=>'长沙',…)不要写成$citys=array(cs=>'长沙',…)。$citys['cs']="长沙"不要写成$citys[cs]="长沙"。否则虽然不会出错，但程序的运行效率将大打折扣。

4. PHP数组的特点

(1) 数组索引既可以是整数，也可以是字符串。如果索引值是整数，则称为索引数组，如果索引值是字符串，则称为关联数组。如果既有整数又有字符串，则称为混合数组。

(2) 数组长度可以自由变化。

(3) 同一数组中各元素的数据类型可以不同，甚至数组元素可以又是数组。例如：

```
$hybrid=array("长沙",0731,true,array("天心区","雨花区","芙蓉区"));
```

输出结果为：

```
Array([0]=>长沙 [1]=>473 [2]=>1 [3]=>Array([0]=>天心区 [1]=>雨花区 [2]=>芙蓉区))
```

7.5.2 访问数组元素或数组

1. 访问数组元素

数组元素也是变量，访问单个数组元素最简单的方法就是通过"数组名[索引]"的形式访问。例如，$i=$citys[3]、echo $citys[1]。也可以使用大括号访问数组元素，如echo $arr{3}。

如果要访问所有数组元素，可以使用foreach语句遍历数组。

2. 添加、删除、修改数组元素

如果给已存在的数组元素赋值，将修改这个数组元素的值，给不存在的数组元素赋值，将添加新的数组元素，而要删除数组元素，一般使用unset()方法。例如：

```
<?  $arr=array(11,22,33,44);
    $arr[0]=66;                   //修改数组元素
    $arr[1]='长沙';               //修改数组元素
    unset($arr[2]);               //删除数组元素
$arr[]=55;                        //添加数组元素
$arr[5]=88;                       //添加数组元素
print_r($arr);     ?>
```

运行结果为：

```
Array([0]=>66 [1]=>长沙 [3]=>44 [4]=>55 [5]=>88)
```

注意：删除的元素索引不会被新添加的数组元素占用。

3. 访问数组

数组名代表整个数组，将数组名赋值给变量能够复制该数组，数组名前加“&”表示该数组的地址，数组同样支持传值赋值和传地址赋值。例如：

```
<?  $citys=array('长沙','衡阳','常德','湘潭');
$urban=$citys;                    //复制数组(传值赋值)
$urban[1]='娄底';                 //修改新数组元素的值
print_r($citys);                  //打印原数组
print_r($urban);                  //打印新数组
                                  //下面为传地址赋值
$loc=&$citys;                     //引用复制数组(传地址赋值)
$loc[1]='郴州';                   //修改新数组元素的值
print_r($citys);                  //打印原数组
print_r($loc);                    //打印新数组
?>
```

输出结果为：

```
Array([0]=>长沙 [1]=>衡阳 [2]=>常德 [3]=>湘潭)
Array([0]=>长沙 [1]=>娄底 [2]=>常德 [3]=>湘潭)
Array([0]=>长沙 [1]=>郴州 [2]=>常德 [3]=>湘潭)
Array([0]=>长沙 [1]=>郴州 [2]=>常德 [3]=>湘潭)
```

可见，引用赋值会使新数组和原数组指向同一个数组的存储区，修改新数组的值，原数组的值也随之改变。

7.5.3　多维数组

创建多维数组同样有两种方法：一是使用 array()函数，二是直接给数组元素赋值。例如要创建一个如下的二维数组：

"玫瑰"	"百合"	"兰花"	
"苹果"	"香蕉"	"葡萄"	"龙眼"

使用 array()函数创建的代码如下:

```
$arr=array(array("玫瑰","百合","兰花"),array("苹果","香蕉","葡萄","龙眼"));
```

由于这个语句没有给索引赋值,所用的默认的索引如下:

[0][0]	[0][1]	[0][2]	
[1][0]	[1][1]	[1][2]	[1][3]

要访问二维数组的元素,可以使用“数组名[索引 1] [索引 2]”的形式访问。例如:

```
echo  $arr[1][2];               //访问数组元素,输出葡萄
$arr[0][3]="茉莉";              //添加数组元素
$arr[1][3]="桂圆";              //修改数组元素
unset($arr[1][0]);              //删除数组元素
```

输出结果为:

```
葡萄 Array([0]=>Array([0]=>玫瑰 [1]=>百合 [2]=>兰花 [3]=>茉莉) [1]=>Array([1]=>香蕉 [2]=
>葡萄 [3]=>桂圆))
```

7.5.4 操作数组的内置函数

PHP 提供了很多操作数组的内置函数,用来对数组进行统计、快速创建、排序等操作。

1. count()函数

count()可返回数组中元素的个数,语法格式为:int count(array arr[, int mode])。例如:

```
<?  $citys=array('长沙','衡阳','常德','湘潭');
echo count($citys);             //输出 4
$arr=array(array("玫瑰","百合","兰花"),array("苹果","香蕉","葡萄","龙眼"));
echo count($arr);               //输出 2
echo count($arr,1);             //输出 9,第 1维 2个,第 2维 7个,共 9个
?>
```

提示:如果数组是多维数组,则 count()函数默认也是统计第一维的元素个数;如果要统计多维数组中所有元素的个数,则可以将 mode 参数的值设置为 1。

2. max()、min()、array_sum()函数

max()和 min()可分别返回数组中最大值元素和最小值元素,而 array_sum()可统计

所有元素值的和。例如：

```
<?  $score=array(70,80,92,60);
echo max($score);               //输出 92
$grade=array('A','C','D','B');
echo min($grade);               //输出 A
echo array_sum($score);         //输出 302
?>
```

3. array_count_values()函数

该函数用于统计数组中所有值出现的次数，并将结果返回到另一数组中。例如：

```
<?  $level=array(2,1,3,1,2,3,2,4,3,1,4);
$tmp=array_count_values($level);
print_r($tmp);           //输出 Array([2]=>3 [1]=>3 [3]=>3 [4]=>2)
?>
```

该函数常用在投票程序中，可以将原数组中的每个值看成是一个投票选项。

4. explode()函数

explode()通过切分一个具有特定格式的字符串而形成一个一维数组，数组中的每个元素就是一个子串。语法为：explode(separator, string[,limit])。例如：

```
<?  $str='湖南 湖北 广东 河南';
$arr=explode(" ",$str);          //通过切分生成数组$arr
print_r($arr);   ?>
```

输出结果为：

```
Array([0]=>湖南 [1]=>湖北 [2]=>广东 [3]=>河南)
```

说明：

(1) explode 的分隔符可以是空格等一切字符，但不能为空字符串""。

(2) limit 参数可限制返回数组元素的个数。

5. implode()函数

implode()使用连接符将数组中的元素连接起来形成一个字符串。它实现了与explode()相反的功能。例如：

```
<?  $grade=array('A','C','D','B');
$link=implode("--",$grade);
echo $link;            //输出 A--C--D--B  ?>
```

6. range()函数

range()可以快速创建一个从参数 start 到 end 的数字数组或字符数组。语法为：

array range(mixed start，mixed end)。例如：

```
<?  $score=range(2,5);          //等价于$score=array(2,3,4,5);
print_r($score);                //输出 Array([0]=>2 [1]=>3 [2]=>4 [3]=>5)
$score=range('D','A');          //等价于$score=array('D', 'C', 'B','A');
?>
```

7. 排序函数

数组排序函数如表7-6所示。注意排序函数中的a表示association,表示排序过程中保持“键值对”的对应关系不变。r表示reverse,表示按降序进行排序。k表示key,表示按照数组元素的“键”进行排序。natsort函数是用“自然排序”的算法对数组arr元素的值进行升序排序,所谓自然排序,是指小数总排在大数前,如img2.gif将排在img10.gif之前。

表7-6 数组排序函数

函　数	排序依据	排序规则	“键值对”是否改变
sort()	元素值	升序	是
rsort()	元素值	降序	是
asort()	元素值	升序	否
arort()	元素值	降序	否
ksort()	索引值	升序	否
krsort()	索引值	降序	否
natsort()	元素值	升序	否
natcasesort()	元素值	升序	否
shuffle()	元素值	随机乱序	是

排序函数示例程序如下：

```
<?  $pic=array('img12.gif','img10.gif','img2.gif','img1.gif','img01.gif');
sort($pic);               //将 sort 依次换成 asort,rsort,arsort
print_r($pic);   ?>
```

输出结果依次为：

```
Array([0]=>img01.gif [1]=>img1.gif [2]=>img10.gif [3]=>img12.gif [4]=>img2.gif)
Array([4]=>img01.gif [3]=>img1.gif [1]=>img10.gif [0]=>img12.gif [2]=>img2.gif)
Array([0]=>img2.gif [1]=>img12.gif [2]=>img10.gif [3]=>img1.gif [4]=>img01.gif)
Array([2]=>img2.gif [0]=>img12.gif [1]=>img10.gif [3]=>img1.gif [4]=>img01.gif)
```

8. array_reverse()和array_unique()函数

array_reverse函数用来对数组元素进行逆序排列,返回逆序后的新数组。array_

unique()函数可删除数组中重复的元素,返回没有重复值的新数组。例如:

```
<? $color=array('a'=>'blue','red','green','red');
$revearr=array_reverse($color);
$uniarr=array_unique($color);
print_r($revearr);     //输出 Array([0]=>red [1]=>green [2]=>red [a]=>blue)
print_r($uniarr);      //输出 Array([a]=>blue [0]=>red [1]=>green)     ?>
```

9. 搜索函数

搜索函数用来检查数组中是否存在某个值或某个键名,假设示例数组为 $color=array('a'=>'blue','red','green','red'),则各搜索函数的功能如表 7-7 所示。

表 7-7　数组搜索函数及功能

函　　数	功　　能	示　　例
in_array(mixed target, array arr)	检查数组中是否存在某个值,返回 true 或 false	in_array('red', $color),返回 true
array_search(mixed target, array arr)	检查数组中是否存在某个值,如果存在则返回其对应的索引值,否则返回 false	array_search('blue', $color),返回 a
array_key_exists(mixed key, array arr)	检查数组中是否存在指定的键,返回 true 或 false	array_key_exists(3, $color),返回 false
array_keys(array arr, mixed search)	返回数组中所有的键名,将其保存到一个新数组中。若指定了 search,则只返回该值对应的键名	array_keys($color),返回 Array([0]=>a [1]=>0 [2]=>1 [3]=>2) array_keys($color,'red'),返回 Array([0]=>0 [1]=>2)
array_values(array arr)	返回数组中所有的值,将其保存到一个新数组中	array_values($color),返回 Array([0]=>blue [1]=>red [2]=>green [3]=>red)

提示: 如果想检查数组中是否含有某个键名,可以用 array_search()函数,如果想获取所有匹配的键名,则应该使用 array_keys()函数,并设置 search 参数。

10. 数组和变量间的转换函数

1) list 函数

list()函数可以用数组中的元素为一组变量赋值,从而通过数组得到一组变量。格式为:

```
void list(var1, var2,…, var n)=array arr
```

list 要求数组 arr 中所有键为数字,并要求数字键从 0 开始连续递增。例如:

```
<? $str='湖南 湖北 广东 河南';
$arr=explode(" ",$str);                //$arr=array([0]=>湖南 [1]=>湖北 [2]=>广东 [3]=>河南)
list($s1,$s2,$s3)=$arr;         //$s1='湖南',$s2='湖北',$s3='广东'
echo $s1."<br>".$s2."<br>".$s3."<br>";      ?>
```

2）extract 函数

extract()函数能利用一个数组生成一组变量，其中变量名为数组元素的键名，变量值为数组元素的值。如果生成的变量名和已有的变量名冲突，则可使用其第二个参数按一组规则来处理。

3）compact 函数

compact 函数利用一组变量返回一个数组，它实现了和 extract 函数相反的功能。数组元素的键名为变量名，数组元素的值为变量值。extract 函数和 compact 函数的示例如下：

```
<? $citys=array("cs"=>"长沙","hy"=>"衡阳",cd=>"常德",xt=>"湘潭");
extract($citys);              //$cs='长沙',$hy='衡阳',$cd='常德',$xt='湘潭'
echo $xt;                     //输出湘潭
$newcitys=compact('cs','cd','xt');          //用变量组成数组
print_r($newcitys);           //输出 array([cs]=>长沙 [cd]=>常德 [xt]=>湘潭)    ?>
```

11. 数组指针函数

每一个 PHP 数组在创建之后都会建立一个“当前指针”(current)，该指针默认指向数组的第一个元素；通过指针函数可获取指针指向的元素值或键名，也可移动当前指针，对数组进行遍历。数组指针函数如表 7-8 所示。

表 7-8 数组指针函数

函 数	功 能
current()	返回当前指针所指元素的“值”
key()	返回当前指针所指元素的“键名”
next()	移动指针使指针指向下一个元素
prev()	移动指针使指针指向上一个元素
end()	使指针指向最后一个元素，并返回当前指针所指元素的值
reset()	使指针指向第一个元素，并返回当前指针所指元素的值
each()	以数组形式返回当前指针所指的元素，该数组有 4 个元素，其中键名为 1 和 value 的元素值为当前元素的值；键名为 0 和 key 的元素值为当前元素的键名

例 7.1 移动数组指针并遍历数组。

利用 next()函数和循环语句可以遍历数组,以实现和 foreach 语句类似的功能。

```
<? $citys=array("cs"=>"长沙","hy"=>"衡阳",cd=>"常德",xt=>"湘潭");
echo key($citys).' '.current($citys).' '.next($citys).' '.next($citys).'<br>';
echo  prev($citys).' '.end($citys).' '.reset($citys).'<br>';
print_r(each($citys)).'<br>';
    reset($citys);
do{
        echo key($citys).'=>'.current($citys);  }
while(next($citys)!==false);        //不要写成 while(next($citys));   ?>
```

输出结果为:

```
cs 长沙 衡阳 常德
衡阳 湘潭 长沙
Array([1]=>长沙 [value]=>长沙 [0]=>cs [key]=>cs)
cs=>长沙 hy=>衡阳 cd=>常德 xt=>湘潭
```

提示:上例中 do…while 语句的循环条件不要简写成 next($citys),因为若某个数组元素的值为空或 0,则值也会被当成 false 处理,导致遇到 0 就会终止循环。

习 题 7

一、练习题

1. 下列哪个 PHP 变量的名称是错误的?( )。

   A. $5-zhao B. $s_Name C. $_if D. $This

2. 语句"echo 'happy'. 1+2 .'345';"输出结果为( )。

   A. 2345 B. happy3345 C. happy12345 D. 运行出错

3. ?:运算符相当于以下哪个 PHP 语句?( )

   A. if...else B. switch C. for D. break

4. 语句"for($k=0; $k=1; $k++);"和语句"for($k=0; $k==1; $k++);"的执行次数分别是( )。

   A. 无限次和 0 B. 0 和无限次 C. 都是无限次 D. 都是 0

5. 如果要提前离开 for 循环,可以使用下面哪个语句?( )

   A. pause B. return C. exit D. break

6. 如果要使程序的运行在循环内跳过后面的语句,直接返回循环的开头,应在循环内使用下面哪个语句?( )

   A. goto B. jump C. continue D. break

7. 对于 for($i=100; $i<=200; $i+=3),循环运行结束后,变量 $i 的值是多少?(　　)

A. 201　　B. 202　　C. 199　　D. 198

8. 下列哪一项代表无穷循环?(　　)

A. for(;;)　　B. for()　　C. foreach(,)　　D. do(1)

9. 数组是通过下列哪一项来区分它所存放的元素的?(　　)

A. 长度　　B. 值　　C. 索引　　D. 维度

10. 在默认情况下,PHP 数组中第一个元素的索引是(　　)。

A. 0　　B. 1　　C. 空字符串　　D. 不一定

11. PHP 规定数组的索引可以为以下哪两种形式?(多选)(　　)

A. 布尔　　B. 浮点型　　C. 整数　　D. 字符串

12. 下列哪一项可以用来访问数组的元素?(　　)

A. ->　　B. =>　　C. ()　　D. []

13. 下列哪些运算符可以用来比较两个数组是否不相等?(　　)

A. +　　B. !=　　C. <>　　D. !==

14. 如果数组 $a=array(0=>5,1=>10), $b=array(1=>15,2=>20), $c=$a+ $b,则 $c 等于下列哪一项?(　　)

A. array([0]=>5[1]=>10[2]=>20)

B. array([0]=>5[1]=>15[2]=>20)

C. array([0]=>5[1]=>[2]=>20)

D. array([0]=>5[1]=>10[2]=>15[3]=>20)

15. 假设 $a=array(0=>'a',1=>'b'), $b=array(1=>'b',0=>'a'),则 $a== $b 和 $a=== $b 的值分别是(　　)。

A. true true　　B. true false　　C. false false　　D. false true

16. 假设 $a=array('a','b','c','d'),则依次调用 next($a);next($a);next($a);prev($a);后,current($a)会返回哪一项?(　　)

A. 'a'　　B. 'b'　　C. 'c'　　D. 'd'

17. 假设 list($x, $y)=array(10,20,30,25),则 $y 的值是(　　)。

A. 10　　B. 20　　C. 30　　D. 25

18. 下列哪个函数可以将数组中的索引和值互相交换?(　　)

A. array_reverse()　B. array_walk()　　C. array_flip()　　D. array_pad()

19. 假设 $a=array(10,25,30,25,40),则 array_sum($a)会返回(　　)。

A. array([0]=>105)　　B. array([0]=>130)

C. 105　　D. 130

20. 假设 $a=range(1,20,5),则 print_r($a)为下列哪一项?(　　)

A. array(1,6,11,16)　　B. array(1,20,5)

C. array(5,10,15,20)　　　　D. array(5,10,15)

21. 假设 $a=array('x','y');,则 $a=array_pad($a,4,'z');,会返回下列哪一项?(　　)

A. array('x','y','z','z')　　　　B. array('z','z','z','z')

C. array('x','x','x','z')　　　　D. array('x','y','z',0)

22. PHP 是________的缩写,PHP 文件中可包含________、________、________三部分的代码。

23. 当把布尔值转换为整型时,true 会转换成________,false 转换成________。当把布尔值转换成字符串时,true 会转换成________,false 转换成________。

24. 已知 $a=8,则执行语句"$b=Sa+=10;"后,$a 的值是________,$b 的值是________。

检测一个变量是否设置需要使用________函数,检测一个变量是否为空需要使用________函数。

25. 对变量进行引用赋值时,引用的变量名前必须加________。

26. 对于用 $arr=array(1,2,array('h'))定义的数组,数组元素'h'的索引值是________,count($arr,1)将返回________。

27. "echo count("abc");"的输出结果是________。

28. 对数组进行升序排序并保留索引关系,应使用的函数是________。

29. 假设网站目录为 E:\news,网站的 admin 目录中的 sh.php 中有包含语句 require "'inc/conn.php';",则应保证文件 conn.php 位于________目录下,如果将该文件包含命令改成"require '/inc/conn.php';",则应保证文件 conn.php 位于________目录下。

30. 假设要输出正确的 HTML 代码,下列 PHP 代码中写法正确的有:________。

(1) <ta<?="b" ?>le border="1">

(2) <ta<?=b ?>le border="1">

(3) <ta<? echo 'b' ?>le border="1">

(4) <p align="<?="right" ?>">段落</p>

(5) <p align='<?="right" ?>'>段落</p>

(6) <p <?="align='right'" ?>>段落</p>

(7) <p <?='align="right"' ?>>段落</p>

(8) <? 'for($i=1;$i<5;$i++)'?>

(9) <? for($i=1;$i<5;$i++) ?>

(10) <? for($i=1;?><? $i<5;$i++) ?>

(11) <% for i=1 to 5 %>

(12) <?="<table border='1'>" ?>

(13) <font size="<?=6?>">天

(14) <style>p{ height:<?=58?>px;}</style>

31. 如果要将一个变量的数据类型由字符串型强制转换成整型，有哪几种方法？

32. 在页面 A 中定义的普通变量 $b 可以在页面 B 中使用吗（页面 A、B 不存在包含关系）？

33. 包含文件操作常用的 4 种函数是什么？各适合应用于哪种场合？

二、编程题

1. 编写 PHP 程序，计算 1～100 之间所有偶数的总和，然后把结果输出出来。

2. 编写程序，在网页上输出一个三角形形式的九九乘法表。

3. 编写程序，使用 while 循环计算 4096 是 2 的几次方，然后输出结果。

4. 编写程序，先声明一个数组{5，8，2，3，7，6，9，1，8，4，3，0}，然后输出数组中最大元素和最小元素的索引值。

5. 先根据原理写出下列程序的运行结果，然后上机验证结果是否正确。

（1）运行结果为：

```
$a="hello";
$b=&$a;
unset($b);
$b="world";
echo $a;
```

（2）运行结果为：

```
$str="true or false;";
if(eval($str))
    echo 1;
else
    echo 0;
```

（3）运行结果为：

```
$n=10;  $nn=100;
$a='$nnn';
$b="$nnn";
$c=$a.$b;
echo $c;
```

（4）运行结果为：

```
$d=3;    $y=1;
while($d>0) {
  $x=($y+1) * 2;
  $y=$x;
  --$d;   }
echo $x;
```

（5）运行结果为：

```
$c=5;
$d=0;
if($c=$d++)
    echo $d;
else
    echo $c;
```

（6）运行结果为：

```
$str='Heng_yang';
$arr=explode('_',$str);
$res=implode('',$arr);
echo $res;
```

6. 写出下列程序的运行结果。

（1）运行结果为：

```
$num=2;
$a=2;    $b=1;
for($i=1;$i<=$num;$i++){
    $s=$s+$a/$b;
    $t=$a;
    $a=$a+$b;
    $b=$t;    }
echo $s;
```

（2）运行结果为：

```
$num=10;
function multiply(){
    $num=$num * 10;
}
multiply();
echo $num;
```

(3）运行结果为：

```
function fun($a){
if($a>1)
  $r=$a * fun($a-1);
else  $r=$a;
return $r;
}
echo fun(3);
```

(4）运行结果为：

```
function rev($var){
  $res="";
for($i=0,$j=strlen($var);$i<$j;
        $i++) {
  $res=$var[$i].$res;
    }
  return $res;
}
$tmp="hengyang";
$res=rev($tmp);
echo $res;
```

(5）运行结果为：

```
$b=20;
$c=40;
$a=$b>$c ? ($c-$b) ? 1:($b-$c)>0: ($b+
$c) ? 0 :$b * $c;
echo $a;
```

(6）运行结果为：

```
$arr=array(1, 1);
for($i=2;$i<20;$i++){
  $arr[$i]=$arr[$i-1]+$arr[$i-2];
}
for($i=0;$i<count($arr);$i++){
    if($arr[$i] %5==0){
        echo $arr[$i];
        break;
    }
}
```

(7）运行结果为：

```
function t($n){
static$num=1;
for($j=1;$j<=$n;$j++) {
   if($j>=4 &&$j<15)
     $num++;
     t($n-$j);
      if($j==20)
        $num--;
     }
     return $num;
}
echo t(5);
```

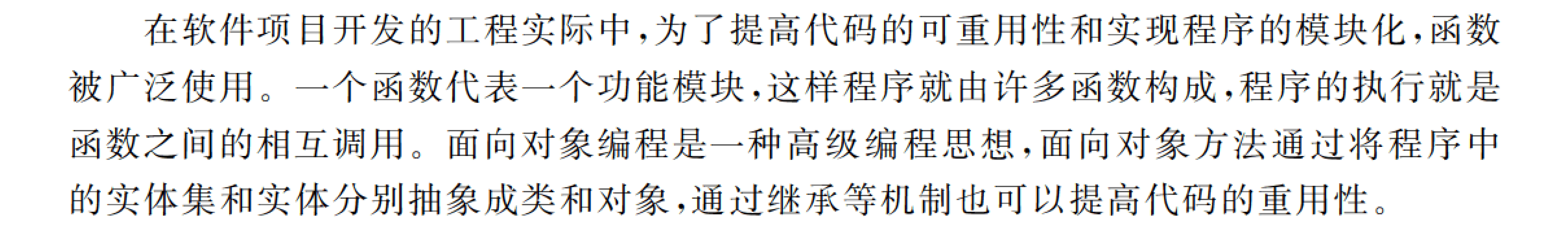
(8）运行结果为：

```
for($i=100;$i<1000;$i++){
    $a=intval($i /100);
    $b=intval($i / 10) %10;
    $c=$i %10;
    if(pow($a, 3)+pow($b, 3)+
    pow($c, 3)==$i &&$i %10==0)
        echo $i;
}
```

(9）运行结果为：

```
function fun($n){
  if($n==3) return 1;
    $t=2 *  (fun($n+1)+1);
  return $t;     }
echo fun(1);
```

第8章 函数和面向对象编程



在软件项目开发的工程实际中，为了提高代码的可重用性和实现程序的模块化，函数被广泛使用。一个函数代表一个功能模块，这样程序就由许多函数构成，程序的执行就是函数之间的相互调用。面向对象编程是一种高级编程思想，面向对象方法通过将程序中的实体集和实体分别抽象成类和对象，通过继承等机制也可以提高代码的重用性。

8.1 PHP的内置函数

PHP提供了大量的内置函数，用于方便开发者对字符串、数值、日期、数组等各种类型的数据进行处理。内置函数无须定义就可使用，如date()就是PHP的一个内置函数。

8.1.1 字符串处理函数

在PHP程序开发中对字符串的操作非常频繁。如用户在注册时输入的用户名、密码以及用户留言等都被当作字符串来处理。很多时候要对这些字符串进行截取、过滤、大小写转换等操作，这时就需要用到字符串处理函数。常用的字符串处理函数如表8-1所示。

表8-1 常用的字符串函数及功能

函　数	功　能	示　例
strlen(string)	返回字符串的长度(中文算两个字符)	strlen("abc8"),返回 4
trim(string[,s])	去掉字符串两端的空格或指定字符	trim(" abcd * "),返回"abcd * "
ltrim(string)、rtrim(string)	去掉字符串左边或右边的空格	ltrim(" abcd * "),返回"abcd * "
substr(string, start, [length])	从字符串的第 start 个字符开始，取长为 length 的子串。如果省略 length，表示取到字符串的结尾，如果 start 为负数表示从末尾开始截取，如果 length 为负数，则表示取到倒数第 length 字符	substr("2010-9-6",5),返回"9-6" substr("2010-9-6",2,4),返回"10-9" substr("2010-9-6",2,-2),返回"10-9" substr("2010-9-6",-3,3) ,返回"9-6"

续表

函　　数	功　　能	示　　例
str_replace(find, replace, string, [&count])	替换字符串中的部分字符,将 find 替换为 replace,如果有参数 count,还可获取替换了多少处	str_replace("AB","*","ABCabc"),返回"*Cabc"
strtr(string, find, replace)	等量替换字符串中的部分字符,将 find 替换为 replace,如果 find 和 replace 长度不同,则只替换两者中的较小者	strtr("Hilla Warld","ial","eo"),返回"Hello World"(i 替换成 e,a 换成 o)
substr_replace(string, replace,start,[length])	从字符串的第 start 个字符开始,用 replace 替换长度为 length 的字符,若省略 length,将替换到结尾	substr_replace("ABCabc","*",3),返回"ABC*" substr_replace("ABCabc","*",3,2),返回"ABC*c"
strtok(string,split)	根据 split 指定的分隔符把字符串分割为更小的字符串	strtok($str," ")
strpos(string, find, [start])	返回子串 find 在字符串 string 中第一次出现的位置,如果未找到该子串,则返回 false,如果有 start 参数,表示开始搜索的位置	strpos("ABCabc","bc"),返回 4 strpos("ABCabc","bc",5),返回 false
strstr(string,search)	返回从 search 开始,字符串的其余部分。如果未找到所搜索的字符串,则返回 false	strstr("ABCabc","ab"),返回"abc"
strspn(string, find, [start])	返回字符串 string 中包含 find 中特定连续字符的数目	strspn("babaadabc","abc"),返回 5
strcmp(str1,str2)	返回两个字符串比较的结果。str1 小于 str2,比较结果为 −1;str1 等于 str2,比较结果为 0;str1 大于 str2,比较结果为 1	strcmp ("ABC","abc"),返回−1 strcmp ("abc","abc"),返回 0 strcmp ("abc","aa"),返回 1
strrev(string)	反转字符串	strrev("Hello"),返回"olleH"
str_repeat(string, repeat)	把字符串重复指定的次数	str_repeat(".",6),返回"......"
nl2br(string)	将 string 中的\n 转换为换行标记 	nl2br("a\nb"),返回"a b"
strip_tags(string,[allow])	去除字符串中的 HTML、XML、PHP 标记	strip_tags("Hello world! "),返回"Hello world!"
chr(number)	返回与指定 ASCII 码对应的字符	chr(13),返回回车符 chr(0x52),返回"R"
ord(string)	返回字符串中第一个字符的值	ord("h"),返回 104
strtolower($str)	字符串转换为小写	strtolower('ABc'),返回 abc

提示：上述字符串函数都严格区分大小写。如果希望不区分则可使用对应的不区分大小写函数(→表示对应关系)：strpos()→stripos()、strstr()→stristr()、str_replace()→str_ireplace()、strcmp()→strcasecmp()，另外 strchr()是 strstr()的别名。

下面介绍 strpos()和 str_replace()函数及其应用实例。

1. strpos()函数

strpos()函数除了有定位子串位置的功能外，还具有查找子串的功能，只要检测其返回值不恒等于 false 即可。

注意：不能用返回值是否等于 0 来判断，因为如果特定子串的位置是第 0 个字符，其返回值也为 0。示例程序如下。

例 8.1 对用户输入的字符串进行检查并过滤掉非法字符。

```
<?
$Patternstr="黄|黑|走私|发票|枪支|东突";              //定义要过滤的非法字符串集
$Pattern=explode("|",$Patternstr);                    //将字符集分割成数组
                                                      //print_r($Pattern);
$inputstr="黑色黄色东突枪支弹药走私物品增值发票";           //假设这是用户输入的字符串
for($i=0;$i<count($Pattern);$i++)    {                //分别对数组中每个字符串进行查找
  if(strpos($inputstr,$Pattern[$i])!==false)    {     //如果找到字符集中的某个字符串
    $outstr=str_replace($Pattern[$i],"",$inputstr);   //将该字符串过滤掉
    $inputstr=$outstr;}        //让输入的字符串等于这次过滤后的字符串
}
echo $outstr."<br>";         ?>
```

程序的输出结果为：色色弹药物品增值。

例 8.2 用字符串函数来判断 E-mail 或 IP 地址的格式是否正确，运行结果如图 8-1 所示。

```
<?  $email="tangsix@163.com";
if(strpos($email, "@") && strpos($email, ".")&& strpos($email, "@")<strpos
($email, "."))
echo "Email 格式正确<br/>";
    //判断 IP 地址是否正确,用到了 explode 函数
$IP="59.51.24.54";
$arr=explode(".",$IP);
if(count($arr)==4)
echo "IP 格式正确,IP 前两位为$arr[0].$arr[1].*.*";      ?>
```

2. str_replace()函数

str_replace()函数除了可替换字符串中的字符外，如果替换后的字符串为空，则能过滤掉被替换字符串中的某些字符，示例程序如下。

例 8.3 对查询关键词描红加粗，运行结果如图 8-2 所示。

```
<?  $content="《Web标准网页设计与 ASP》";          //假设这是待查询信息
$find="网页设计";                                  //假设这是查询关键词
$out=str_ireplace($find,"<b style='color:red'>$find</b>",$content);
echo $out."<br>";  ?>
```



图 8-1　例 8.2 的运行结果



图 8-2　例 8.3 的运行结果

8.1.2　日期和时间函数

在动态网站中，经常需要获取当前的日期时间信息，比如在论坛中要记录发言的日期和时间等，使用 PHP 提供的日期函数能方便地获取日期时间。

1. date()函数

date(string,[stamp])是最常用的日期时间函数。用来返回或设置当前日期或时间。例如：

```
echo date("Y-m-d");              //输出 2013-04-23
echo date("y年 m月 d");          //输出 13年 04月 23
echo date("h:i:s");              //输出 10:44:46
```

其中，Y、m、d 等是 date 函数 string 参数中的格式字符，常见的格式字符如表 8-2 所示。除了格式字符外的字符都是普通字符，它们将按原样显示，如“年”、“-”等。

表 8-2　date()函数的格式字符及其说明

格式字符	说　明	格式字符	说　明
Y	以 4 位数显示年	H	以 24 小时制显示小时(会补 0)
y	以 2 位数显示年	G	以 24 小时制显示小时(不补 0)
m	以 2 位数显示月(会补 0)	h	以 12 小时制显示小时(会补 0)
n	以数字显示月(不补 0)	g	以 12 小时制显示小时(不补 0)
M	以英文缩写显示月	i	以 2 位数显示分钟(会补 0)
d	以 2 位数显示日(会补 0)	s	以 2 位数显示秒(会补 0)
j	以数字显示日(不补 0)	t	显示该日期所在的月有几天，如 31
w	以数字显示星期(0～6)	z	显示该日期为一年中的第几天
D	以英文缩写显示星期	T	显示本地计算机的时区
l	以英文全称显示星期	L	判断是否为闰年，1 表示是

提示：

(1) date()函数也可带有两个参数，此时用来设置时间。第二个参数必须是一个时间戳，它将使 date()返回时间戳设置的时间。例如，date('Y-m-d', 0)将返回 1970-01-01。

(2) PHP 解析器默认采用格林尼治时间，使得调用时间函数与实际时间相差 8 小时。为此，需要设置 PHP 的时区，打开 php.ini 文件，将“;date.timezone”修改为“date.timezone=PRC”即可。

2. getdate()函数

getdate()函数也能返回当前的日期时间，但它会返回各种时间字段到数组中。例如：

```
<? $today=getdate();
      print_r($today);              //$today是 getdate()函数返回的数组
echo "$today[mon]月 $today[mday]日";    //mon 和 mday是数组元素的索引
?>
```

其中，print_r()是用于递归打印数组或对象的语句，可以将数组整体输出。运行结果为：

```
Array([seconds]=>58 [minutes]=>8 [hours]=>12 [mday]=>26 [wday]=>5 [mon]=>4 [year]=>2013 [yday]
=>115 [weekday]=>Friday [month]=>April [0]=>1366974538) 4月 26日
```

3. time()函数

time()函数会返回当前时间的时间戳。所谓时间戳，是指从 1970/1/1 日 0:0:0 到指定日期所经过的秒数。例如，当前时间为 2013-04-28 11:58:17，则 time()返回的时间戳是 1367146697。因此利用 time()可对时间进行加减。示例程序如下，运行结果如图 8-3 所示。

```
<? $nextWeek=time()+(7 * 24 * 60 * 60);     //1周=7天 * 24小时 * 60分 * 60秒
echo '现在是：'.date('Y-m-d')."<br>";
echo '下一周是：'.date('Y-m-d',$nextWeek);    ?>
```



图 8-3 time()函数示例

4. mktime()函数

mktime()函数会返回自行设置的时间的时间戳。与 date()函数结合使用可以对日期进行加减运算及验证。其语法如下：int mktime(时,分,秒,月,日,年)，例如：

```
echo date("Y-m-d",mktime(0,0,0,12,36,2012));
```

表示设置时间为 2012 年 12 月 36 日，则 mktime()会自动校正时间越界，输出结果为：2013-01-05。如果要在今天日期的基础上加 12 天，可以使用：

```
echo date("Y-m-d",mktime(0,0,0,date(m),date(d)+12));
```

输出结果为：

2013-05-08(注：系统当前日期为 2013-04-26)

上述代码中省略了年的参数，因为 mktime()函数的参数可以按照从右至左的顺序省略，任何省略的参数都会被设置为当前时间值。

5. strtotime()函数

strtotime()函数可将日期时间(英文格式)解析为时间戳。其功能相当于 date()函数设置时间的逆过程。date()函数(带有 2 个参数时)可以将时间戳设置为时间，而 strtotime()是将时间解析为时间戳。

```
<? echo strtotime("now");                    //输出当前时间的时间戳,如：1367148939
echo strtotime("+5 hours");                  //输出加 5 小时后的时间戳
echo date('Y-m-d',strtotime("+1 week"));     //利用返回的时间戳设置时间
echo strtotime("+1 week 3 days 7 hours 5 seconds");  ?>
```

可见，使用 strtotime()函数也可用来对时间进行加减。

6. checkdate()函数

checkdate(月，日，年)函数可判断参数指定的日期是否为有效日期。如果是，就返回 true，否则返回 false。例如 checkdate(10，3，2014)返回 true，因为 2014/10/3 日是存在的。而 checkdate(13，3，2012)返回 false。checkdate()经常用来对用户输入日期的合法性进行检查。

8.1.3 检验函数

检验函数用来检查变量是否定义、是否为空、获得变量的数据类型、取消变量定义等。

1. isset()函数

isset($var)函数用来检查变量 $var 是否定义。该函数参数为变量名(带 $ 号)，如果变量已经定义，并且其值不为 NULL，则返回 true，否则返回 false。

```
<? echo isset($test);        //返回 false,输出空字符串
   $test=null;
   echo isset($test);        //仍然返回 false  ?>
```

通俗地说，如果有这个变量，则 isset($var)返回 true，否则返回 false。

2. empty()函数

empty()函数用来检查变量是否为空。所谓变量为空，包括两种情况：一是变量未定义；二是变量的值为""、0、"0"、NULL、FALSE 以及空数组或者没有任何属性的对象等。例如：

```
<? $var=0;
echo empty($var);            //变量值为 0,返回 1
echo isset($var);            //变量存在,且值不为 null,返回 1
echo empty($str);            //变量不存在,返回 1  ?>
```

因此,如果要检测变量是否定义,尽量用 isset()方法。

3. unset()函数

unset($var)函数用来取消变量 var 的定义。该函数的参数为变量名,函数没有返回值。需注意的是,如果在某个自定义函数中用 unset()取消一个全局变量,则只是局部变量被取消,而在调用环境中的变量仍将保持调用 unset()之前一样的值。例如:

```
<? $foo='alive';
function destroy_foo() {
    global$foo;
    unset($foo);             //在函数中删除变量$foo,实际上只是局部变量被删除
}
destroy_foo();
echo $foo;                   //仍将输出 alive
unset($bar[1]);              //unset 也能删除数组元素
unset($foo1,$foo2,$foo3);    //同时删除多个变量  ?>
```

4. gettype()函数

gettype()函数用来返回变量或常量的数据类型,返回值包括 integer、double、string、array、object、unknown type 等。其语法格式为:string gettype (mixed var)。例如:

```
<? $foo='bar';
echo gettype($foo).'<br>';       //输出 string
$bar=array("aa",12,true,2.2,"test",50);
echo gettype($bar[1]);           //输出 integer
?>
```

虽然 gettype()函数可用来获取数据类型,但由于 gettype()函数在内部进行了字符串的比较,所以它的运行速度较慢。建议使用下面介绍的 var_dump()函数和 is_ * ()函数来代替。

5. var_dump()函数

var_dump()函数用来返回变量或常量的数据类型和值,并将这些信息输出。例如:

```
<? $a=3.1; $b='天涯';
var_dump($a,$b);                 //输出$a、$b的数据类型和值
$c=array(1, '2', array("a", "b", "c"));
var_dump ($c);                   //输出$c的数据类型和值
?>
```

输出结果为：

```
float(3.1) string(4) "天涯" array(3) { [0]=>int(1) [1]=>string(1) "2" [2]=>array(3) { [0]=>string
(1) "a" [1]=>string(1) "b" [2]=>string(1) "c" } }
```

在调试程序时，经常使用该函数查看变量或常量的值、数据类型等信息。

6. is_*()系列函数

is_*()系列函数包括 is_string()、is_int()、is_float()、is_bool、is_null()、is_array()、is_object()、is_numeric()、is_resource()、is_integer()、is_long()、is_real()等。它们用来判断变量是否为某种数据类型。如果是，则返回 true，否则返回 false。例如，is_string()可以判断变量是否为字符串数据类型，is_int()判断变量是否为整型，而 is_numeric()判断变量是否为数字或由数字组成的字符串。例如：

```
<? $a=3.1;
echo is_float($a);              //返回 true
$b='13307473544';               //$b是由数字组成的字符串
echo is_numeric($b);            //返回 true  ?>
```

7. settype()函数

settype()函数可以进行强制数据类型转换。转换规则遵循表 7-5 的规定。其语法格式为：int settype(string var, string type)，参数 type 为下列的类型之一：integer、double、string、array 与 object。例如：

```
<? $a=3.1;
settype($a, integer);           //将变量$a转换成整型
echo $a;                        //输出 3
$b="false";
settype($b, bool);              //将变量$b转换成布尔型
echo $b;                        //返回 true   ?>
```

8. eval()函数

eval()函数可以动态执行函数内的 PHP 代码，该函数的参数是一个字符串，eval()会试着执行字符串中的代码。示例代码如下：

```
<? eval('$a=5+3;');             //执行赋值语句
   echo $a;                     //输出 8
   eval('var_dump($a);');       //输出 int(8)  ?>
```

虽然 eval()函数非常好用，但是，eval()函数执行代码时效率是十分低的。并且，eval()容易产生安全性问题，在获取表单中用户输入的数据时，应过滤这些数据中的 eval 关键词，因为它允许用户去执行任意代码，这是很危险的。

8.1.4 数学函数

数学函数的参数和返回值一般都是数值型，常用的数学函数如表 8-3 所示。

表 8-3 常用的数学函数及其功能

函　数	功　能	示　例
round(val[,int precision])	返回按指定位数四舍五入的数值，如果省略 precision，则返回整数	round(3.41)，返回 3 round(3.45,1)，返回 3.5
ceil(val)	返回大于并最接近 val 的整数	ceil(3.45)，返回 4
floor(val)	返回小于并最接近 val 的整数	floor(3.45)，返回 3
intval(val)	返回 val 的整数部分	intval('3.6a')，返回 3 intval(3.6)，返回 3
abs(num)	返回 num 的绝对值	abs(－3.43)，返回 3.43
sqrt(num)	返回数 num 的平方根	sqrt(16)，返回 4
pow(base,exp)	计算次方值，base 为底，exp 为幂	pow(2,3)，返回 8
log(num[,base])	计算以 e 为底的对数	log(10)，返回 2.3025…
exp(num)	返回自然对数 e 的幂次方	exp(10)，返回 22026.…
rand(int min,int max)	返回 min 到 max 之间的伪随机数	rand(2,9)，返回 2～9 之间的整数
srand(int seed)	播下随机数发生器种子	已被淘汰，不建议使用
int getrandmax (void)	返回调用 rand()可能返回的最大值	
sin(arg)等三角函数	包括 sin()、cos()、tan()等	sin(pi()/6)，返回 0.5
max (num1, num2, …, numn)	返回若干个参数中的最大值	max(2,3,3.5)，返回 3.5
min (num1, num2, …, numn)	返回若干个参数中的最小值	min(2,3,3.5)，返回 2
decbin(num)	十进制数转换为二进制	decbin(6)，返回 110
bindec(num)	二进制数转换为十进制	bindec(11)，返回 3
dechex	十进制数转换为十六进制	dechex(13)，返回"d"
base_convert(num,from,to)	在任意进制之间转换数字	base_convert('1a',16,10)，返回"26"
number_format (num, preci,[point],[sep])	格式化数字字符串	number_format(3.142,2)，返回"3.14" number_format(1314.5205,3,".", " ")，返回"1 314.521"

8.2　自定义函数及调用

除了直接调用 PHP 内置函数完成某些功能外，用户还可以自己设计函数，来实现某种特殊功能，这称为自定义函数。使用自定义函数包括函数的定义和函数的调用两个步骤。

8.2.1　函数的定义

函数是一个可重用的代码块，用来完成某个特定功能。每当需要反复执行一段代码时，可以使用函数来避免重复书写相同代码。不过，函数的真正威力体现在，函数就像一台机器（见图 8-4），这台“机器”可以接收一些数据作为输入（通过函数的参数），进行加工后再把执行的“结果”输出（通过 return 语句）。函数可以有 0 到多个参数，但只能有一个输出。用户设计函数的第一步就是要想清楚函数的输入和输出。

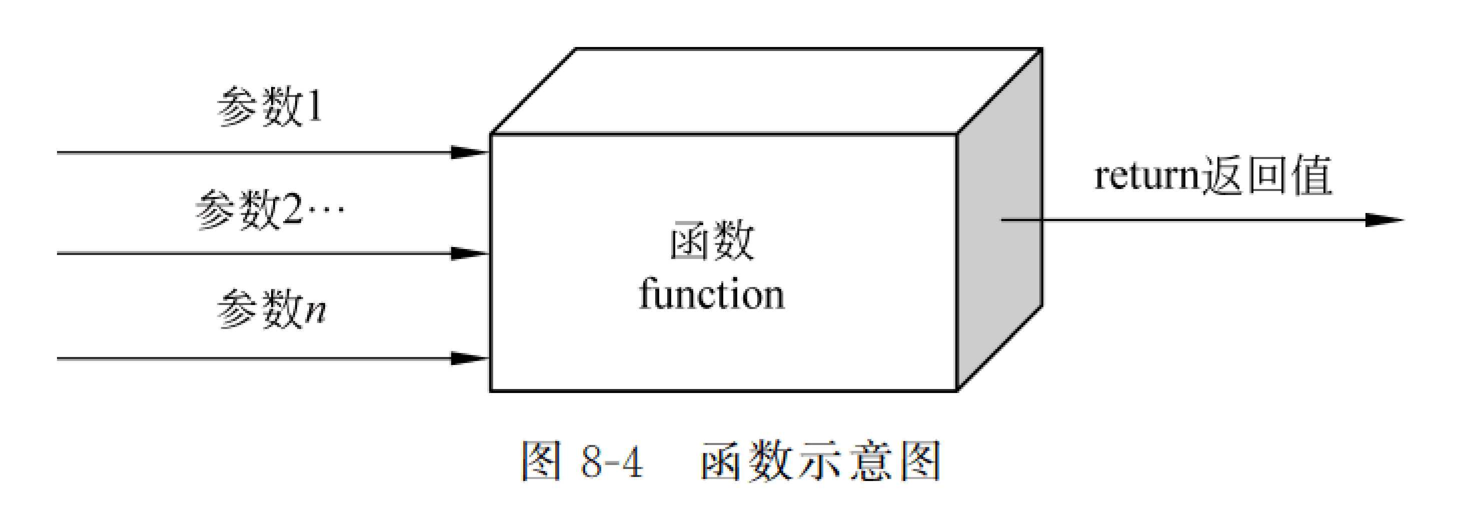


图 8-4　函数示意图

定义函数的语法如下：

```
function 函数名 ([形参 1, 形参 2, …, 形参 n])  {
    函数体
    [return 返回值]
}
```

其中，function 是 PHP 定义函数的关键字，函数名是自定义函数的名称，必须符合变量的命名规则。参数是函数的输入接口，函数通过参数接收“外部”数据。函数体是函数的功能实现。return 语句用来返回函数的执行结果，如果不需要返回结果，则可以没有 return 语句。

下面是一个求两个数和的函数，该函数的输入是两个数，输出是两数的和。代码如下：

```
<?  function sum($a,$b){
      $c=$a+$b;
      return $c;  }            //不要写成 echo $c;
    echo sum(3,4);             //调用函数     ?>
```

提示：在函数体内一般不要有 echo 之类的输出语句，因为大多数时候只是希望将函数返回的值传递给其他程序，而并不需要输出到页面上。一个函数应该是一个单一功能模块，而不应该把函数的特定功能与输出功能混在一起。函数输出应通过 return 语句返

回值实现。

8.2.2 函数的调用

要执行函数内的代码,必须调用函数。函数调用有 3 种方式。即函数调用语句、赋值语句和函数嵌套调用。

1. 函数调用语句

如果函数没有返回值(无论是否有参数),则可使用函数调用语句调用函数,形式为:

```
函数名([实参 1,实参 2,…,实参 n]);
```

例如下面两个程序通过调用自定义的 hello()函数,来打印一行字符:

```
<?
function hello(){
  echo "********************";
}
hello();      //调用无参函数
?>
```

```
<?
function hello($n,$star){
     for($i=0;$i<$n;$i++)
         echo $star;
}
hello(8,'&');          //调用有参函数    ?>
```

例 8.4 设计函数判断手机号码格式是否正确。

```
<?  function isTel($tel)   {
        if(strlen($tel)==11 && is_numeric($tel))
            echo  "手机号码格式正确";
        else
            echo "格式不正确,请重新输入";  }
    isTel("13388888888");         //调用有参函数     ?>
```

2. 赋值语句调用函数

如果函数有返回值,通常使用赋值语句将函数的返回值赋给一个变量。形式为:

```
变量名=函数名([实参 1,实参 2,…,实参 n]);
```

使用赋值语句调用函数的例子包括例 8.5、例 8.6 和例 8.7,注意这些函数都有返回值。

例 8.5 限制输出字符串的长度。

函数 Trimtit()的功能是:如果输入的字符串 $tit 长度大于指定的长度 $n,则返回截取的指定长度字符串并加"…",如果长度小于等于指定长度,则返回原字符串。

```
<?  function Trimtit($tit,$n) {            //注意函数的输入为两个类型不同的参数
if(mb_strlen($tit,'GB2312')>$n)
        return   mb_substr($tit,0,$n,'GB2312')."…";    //返回函数值
    else
        return $tit;                                    //返回函数值
```

```
    }
$str="航空母舰辽宁舰 2012 年完成舰载机着舰";          //测试字符串
$out=Trimtit($str,14);                                //调用函数
echo $out;              //输出:航空母舰辽宁舰 2012 年完成…
?>
```

说明:

(1) 函数的参数类型可以各不相同,如上例中的 $ tit 是字符串,而 $ n 是数值型。

(2) 函数体中只有一条 return 语句会被执行,return 语句以后的函数代码将不会被执行。

(3) mb_strlen()和 mb_substr()分别是 strlen()和 substr()的中文字符的版本,这两个函数都必须带有指定编码类型的参数,如'GB2312'。如果处理的字符串中有中文,一定要用这两个函数,因为 substr()不仅会把中文当成 2 个字符,在处理中文字符时还会产生乱码。

例 8.6　替换特殊字符为字符实体。

有时用户在表单中提交了一段字符串,这段字符串中可能有回车、空格等特殊字符,由于浏览器会忽略代码中的回车、空格等字符,导致这些格式丢失,因此有必要将它们用字符实体替代,使这些格式在浏览器中保留下来,示例程序如下,运行效果如图 8-5 所示。

```
<?
function myReplace($str){
    $str=str_replace("<","&lt;",$str);          //替换<为字符实体 &lt;
    $str=str_replace(">","&gt;",$str);          //替换>为字符实体 &gt;
    $str=str_replace(chr(13),"<br>",$str);      //替换回车符为换行标记<br>
    $str=str_replace(chr(32)," ",$str);    //替换空格符为字符实体  
      return $str;                              //返回函数值
}
$str="<font color='red'>abc</font>";            //测试字符串
echo $str.'<br>';
echo myReplace($str);     ?>
```

提示:PHP 提供了内置函数 htmlentities()可以完成例 8.6 中 myReplace 函数的功能,但是 htmlentities()不会将空格替换成字符实体“ ”。



图 8-5　例 8.6 运行效果

由于函数的返回值只能有 1 个,如果希望函数返回多个值,可以让函数返回一个数组。return 语句返回数组的语法如下,示例代码见例 8.7。

```
return 数组名
```

例 8.7　设计一个函数,输入是一个整数,输出是这个整数各位上的数字。

由于本例中函数的返回值有多个,因此让函数返回一个数组,该数组保存了输入的整数各个位上的数字。

```
<? function aval($num){              //aval()用来求$num各位上的数字
      for($i=0;$num>=1;$i++){
         $arr[$i]=$num%10;           //对 10取余得到个位数
         $num=$num/10;     }         //除以 10后十位数变成个位数
      return $arr;                   //返回数组$arr
   }
   print_r(aval(54262));             //调用函数,将返回各个数位上的数字    ?>
```

输出结果为:

```
Array([0]=>2 [1]=>6 [2]=>2 [3]=>4 [4]=>5)
```

3. 函数的嵌套调用

函数可以嵌套调用,即把函数调用作为另一函数的参数。例如:

```
<? function sum($a,$b){
   return $a+$b;   }
   echo sum(7,sum(3,5));             //函数作为另一函数的参数调用 ?>
```

例 8.8 过滤字符串中的 HTML 标记。

有时需要把文本中的 HTML 标记都过滤掉,过滤的思路是:首先找到第 1 个 HTML 标记的开始和结束位置("<"和">"),将"<"左边的字符与">"右边的字符连接在一起,这样就去掉了第 1 个 HTML 标记,再把过滤后的字符串赋值给原字符串,进行下次过滤,直到文本中找不到 HTML 标记为止。

```
<?            // right 函数:截取字符串$s右边的$n个字符
function right($s,$n) { return $n? substr($s, -$n): ''; }
function noHtml($str){                         //noHTML函数:去除字符串$str中的 HTML标记代码
while (strpos($str,'<')!==false||strpos($str,'>')!==false)     {
                                               //如果字符串中有"<"或">"
    $begin=strpos($str,'<');                   //找到"<"符的位置
    $end=strpos($str,'>');                     //找到">"符的位置
    $length=strlen($str)-$end-1;               //">"符右边的字符串长度
            //将"<"符左边的字符串和">"符右边的字符串连接在一起
    $filterstr=substr($str,0,$begin) . right($str,$length);
                                               //在函数体内调用另一函数
   $str=$filterstr;      //把一次过滤后的字符串赋给原字符串,以便进行下次过滤
}
return $str;                                   //返回函数值
}
$str="<font size=9>abc</font>";                //测试字符串
echo noHtml($str);                             //输出结果为"abc"
?>
```

实际上,PHP 提供了内置函数 strip_tags()可实现 noHtml()函数的功能。

8.2.3　传值赋值和传地址赋值

函数的参数赋值有两种方法，即传值赋值和传地址赋值。

1. 传值赋值

默认情况下，函数的参数赋值采用传值赋值方式，即将实参值拷贝给形参值。这种赋值方式是单向的，即实参将值赋给形参，但调用结束后形参不会将值赋给实参，因此函数调用结束后实参的值不会发生改变。例如：

```
<?  function add1($val){
        $val++;
        return $val;    }
    $age=18;
    echo add1($age).' ';
    echo add1($age).' ';
    echo $age;
```

运行结果为

```
19  19  18  ?>
```

上述程序的执行过程是：

（1）函数只有在被调用时才会执行。因此，程序执行的第一条语句是“$age=18;”，PHP 预处理器为 $age 分配第一个存储空间。

（2）执行语句 echo add1($age).' ';，此时自定义函数 add1()被调用，PHP 预处理器为函数参数 $val 分配存储空间，将实参值 18 拷贝给 $val。

（3）$val 进行加 1 运算，使 $val 的值为 19，但 $age 的值仍为 18。

（4）函数调用结束时，PHP 预处理器回收函数调用期间分配的所有内存，此时 $val 消失。

（5）第二次调用函数时，又将 $age 的值拷贝给 $val，因此 $val 的初始值仍为 18。

2. 传地址赋值

函数的参数也可以使用传地址赋值，即将一个变量的“引用”传递给函数的参数。和变量传地址赋值一样，在函数的参数名前加“&”就能实现传地址赋值。示例代码如下：

```
<?  function add1(&$val){
        $val++;
        return $val;  }
    $age=18;
    echo add1($age).' ';       //注意传地址赋值时，函数参数不能是常量
    echo add1($age).' ';
    echo $age;
```

运行结果为

```
19 20 20    ? >
```

上述程序的执行过程是：

(1) 程序执行的第一条语句是“＄age＝18;”,PHP 预处理器为＄age 分配第一个存储空间。

(2) 程序执行到“echo add1(＄age).' ';”,此时自定义函数 add1()被调用,PHP 预处理器为函数参数＄val 分配存储空间,由于这里是传地址赋值,形参＄val 和变量＄age 都指向同一个变量值 18 的地址。因此＄val 的值变为 18。

(3) 程序执行到“＄val＋＋;”时,形参＄val 修改地址中的值为 19,由于变量＄age 也指向该地址,因此变量＄age 的值也变为了 19。

(4) 函数调用结束时,PHP 预处理器回收函数调用期间分配的所有内存,此时＄val 消失。但函数外变量＄age 的值不会改变,仍然为 19。

(5) 第二次调用时,＄val 又会修改＄age 指向地址中的值,使＄age 的值变为 20。

可见,使用传值赋值的方式为函数参数赋值,函数无法修改函数体外的变量值;若使用传地址的方法为函数参数赋值,则函数可以修改函数体外的变量值。

但无论使用哪种赋值方式,函数参数(或函数体内变量)的生存周期是函数运行期间,若要延长函数体内变量的生存期,需使用 static 关键字;函数参数(或函数体内变量)的作用域是函数体内有效,若要扩大函数体内变量的作用域,需使用 global 关键字。

8.3 面向对象编程

面向对象编程(Object Oriented Programming,OOP)是一种编程思想,目前在大型应用软件开发中应用非常广泛。

面向对象的程序由对象(object)构成。把所有的对象都划分成类(class),每个类定义了一组静态的属性和动态的方法。对象之间通过传递消息互相联系,驱动整个系统来运转。类是具有相同或相似的结构、操作与约束关系的对象组成的集合。对象是对某个类的具体化实例,每个类都是具有某些共同特征对象的抽象。

8.3.1 类和对象

在现实中,任何一个具体事物都可以看作一个对象,例如一个人、一辆汽车、一场电影等都是对象。对象包含属性和方法。将张三这个人看作对象,则该对象具有下列属性和方法。

```
对象:张三  {
    属性:姓名、性别、身高、体重、年龄等;
    方法:吃饭、走路、说话等;
}
```

对于对象的属性,我们可以用变量来描述,例如,＄age＝33 表示张三的年龄是 33 岁。对于对象的方法,则可以用函数来定义,例如,function walk(){…}用来描述走路。

由于现实中很多对象都属于同一类，因此还可以把同一类的对象看成一个类，例如，所有的人都属于“人”类。对象可看成是类的一个实例，例如，张三是“人”类的一个实例。因此定义对象前都要先定义类。

1. 类的定义

在PHP中，使用class关键字可以定义一个类。语法格式为：

```
class 类名{
    定义成员变量
    定义成员函数
}
```

可见，类实际上就是一组静态属性和动态方法的集合，将它们封装在一起就形成了一个类。例如，要定义一个类Mystr，代码如下：

```
<?  class Mystr{                    //定义 Mystr 类，注意类名后面没有小括号
        var $str;
        function output(){
            echo 'Hello PHP';   }
    }  ?>
```

说明：在类定义中，使用关键字var来定义成员变量。在定义成员变量时，也可直接对它赋值。

类成员变量又可分为两种：一种是公有变量，用关键字public或var定义；另一种是私有变量，用关键字private定义。公有变量可以在类的外部被访问，它是类与其他类或用户交流的接口。用户可通过公有变量向类中传递数据，也可以通过公有变量获取类中的数据。私有变量在类的外部无法访问，以保证类的设计思想和内部结构并不完全对外公开，这就是面向对象中的封装性。

下面是定义公有变量和私有变量的例子(8-1.php)。

```
class userInfo{
    public $userName;
    private $pwd;
    function output(){
      echo $this->userName;   }
}
```

在类userInfo中，使用公有变量来保存用户名，使用私有变量来保存用户密码。

说明：

(1) 类一旦定义后，系统会自动为其创建一个$this的伪变量，代表类自身。

(2) 如果类的成员函数中要访问类中的变量或其他函数，必须使用“$this->变量名”或“$this->函数名”访问。例如，$this-> userName。不能简单使用$userName来访问，也不能写成$this-> $userName，更不能写成userInfo-> userName。

(3) 如果要在类外面访问类中定义的变量和方法,必须先创建该类的对象,然后用"对象名->变量名"或"对象名->方法名"来访问。

(4) "->"是 PHP 中的成员选择运算符。该运算符表示右边的变量或函数隶属于左边的类或对象。

注意:区分"->"和"=>","=>"是初始化数组元素时分隔"键"和"值"的符号。

2. 构造函数和析构函数

在定义类时可以在类中定义一个特殊的函数——构造函数,用来执行一些初始化的任务,比如对属性赋初值等。PHP 规定构造函数的名称必须为"__construct"。

例如,在 userInfo 类中定义一个构造函数(8-2.php)。

```
class userInfo{
    public $userName;
    private $pwd;
    function __construct(){            //定义构造函数
        $this->userName='Admin';       //为类中的变量赋初值
        $this->pwd='123';
    }
    function output(){
        echo $this->userName;     }
}
```

说明:

(1) 构造函数名"__construct"是以两个下划线开头。

(2) 构造函数不能被主动调用,例如"对象名->__construct()"是错误的。只有在使用关键字 new 创建对象时,系统才会自动调用构造函数。

与构造函数相对应的是析构函数,析构函数会在某个对象的所有引用被删除或者对象被销毁时执行。也就是说,如果定义了析构函数,则对象在销毁前会调用析构函数。

PHP 规定析构函数的名称为"__destruct()",析构函数不能带有任何参数。

3. 定义对象

对象是类的实例,可以使用 new 关键字来创建对象。定义一个类 userInfo 的对象 $user 的代码如下:

```
$user=new userInfo();
```

则 $user 就是一个对象(类型为 object 的变量),定义了对象后,就可使用对象来访问类中的成员变量或成员方法。例如:

```
$user=new userInfo();
echo $user->userName;          //访问类中的变量
$user->output();               //访问类中的函数
```

注意：

(1) 如果类中的构造函数包含参数，则在创建对象时，也需要提供相应的参数。

(2) 对象只能访问类中的公有变量和函数，如果试图访问类中的私有变量或函数，如 echo $user->pwd，则程序会出错，提示不能访问私有属性。

下面是一个定义类和对象的综合实例(8-3.php)。

```
<? class Person{
    var $name;                  //人的名字
    var $sex;                   //人的性别
    var $age;                   //人的年龄
    function say($word)         //人有说话的方法
        {echo $this->name.'对你说:'.$word;}
    function run($step)         //人有走路的方法
        {echo "<br>然后走了".$step."步";}
    }
$p1=new Person();               //创建类 person 的对象$p1
$p1->name="张三";               //设置对象的属性,形式为"对象名->属性名"
$p1->say('您好');               //访问对象的方法,形式为"对象名->方法名"
$p1->run(5);
?>
```

输出结果为：

```
张三对你说:您好
然后走了 5步
```

4. 操作符“::”

相比伪变量 $this 只能在类的内部使用，操作符“::”更加强大。它可以在没有声明对象的情况下直接访问类中的变量或方法。例如，下面的代码可在类外访问 Person 类的方法。

```
Person::run(8);                 //将其放在 Person类代码的外部,将输出"然后走了 8步"
```

操作符“::”可用于访问静态变量、静态方法和常量，还可用于覆盖类中的成员变量和方法。其语法格式为：

```
关键字 :: 变量名/方法名/常量名
```

其中关键字可以分为以下三种情况。

- 类名：用来调用本类中的变量、常量和方法。
- self：用来调用当前类中的静态成员和常量。
- parent：用来调用父类中的变量、常量和方法。

5. instanceof 关键字

instanceof 关键字用来检测某个对象是否属于某个类，它返回一个布尔值。例如：

```
echo $p1 instanceof Person                    //返回 true
```

8.3.2 类的继承和多态

面向对象的三个基本特性是：封装、继承和多态。封装是指把客观事物的属性和方法封装在一起，形成抽象的类。继承和多态的概念如下。

1. 继承

继承是指子类可以继承一个或多个父类的属性和方法，并可以重写或添加新的属性或方法。通过对已有类的继承，可以逐步扩充类的功能。继承的这些特性简化了对象和类的创建，增加了代码的可重用性。

例如，要设计三个类："动物"类、"人"类和"学生"类，则可以先定义动物类，将动物类作为父类，人类作为子类，通过继承动物类的一些属性和方法就可以简化人类的设计，并可以添加人类的新属性和方法(比如国籍、说话等)。同样地，学生类又可看成是人类的子类。

在 PHP 中，用 extends 关键字可实现类的继承。语法格式为：

```
class 子类名 extends 父类名
{    定义子类的成员变量
     定义子类的成员函数
}
```

提示：PHP 不支持多重继承，即一个子类不能有多个父类。

下面创建了一个类 Students，并使它继承于类 Person，代码如下(8-4. php)：

```
<?
class Person{                                   //定义父类
    function __construct($name,$sex){           //定义构造函数
        $this->name=$name;
        $this->sex=$sex;
        }
    function say(){                             //定义说话的方法
        echo '我叫：'.$this->name;
        echo '性别：'.$this->sex.'<br>';
        }
}
class Students extends Person{                  //定义子类并继承父类
    public $school;
    function study($scholl){                    //定义上学的方法
        echo '我在'.$scholl.'上学';
        }
}
$student=new Students('小新','男');             //创建一个子类的对象
```

```
$student->say();                         //调用父类的方法
$student->study('石鼓书院');             //调用子类的方法
?>
```

运行程序，输出结果为：

```
我叫：小新 性别：男
我在石鼓书院上学
```

说明：程序中子类 Students 通过继承父类 Person，调用了父类中的方法和属性，如显式调用了父类中的 say()方法，通过创建对象隐式调用了父类中的构造方法。同时子类也可调用自己定义的方法 study()。

2. 多态

多态好比请大家吃某种食物，有的人可能用筷子吃，有的人可能用勺子吃，还有的人用叉子和勺子一起吃。虽然是同一种方法，但调用时却产生了不同的形态，这就是多态。

在面向对象中，多态指多个函数使用同一个名字，但参数个数、参数数据类型不同。调用时，虽然方法名相同，但会根据参数个数或者类型自动调用对应的函数。

多态可通过继承或接口来实现。下面是一个通过继承实现多态的例子(8-5.php)。

```
<?
class Person{                                //定义父类
    function __construct($name,$sex){        //定义构造函数
        $this->name=$name;
        $this->sex=$sex;
    }
}
class Students extends Person{               //定义 Person 的子类 Students
        public $school;
        function study(){                    //定义上学的方法
            echo '我在上学<br>';    }
}
class dxs extends Students{                  //定义 Students 的子类 dxs
function study(){                            //定义上学的方法
        echo $this->name.'在读大学<br>';    }
}
class xxs extends Students{                  //定义 Students 的子类 xxs
function study(){                            //定义上学的方法
        echo $this->name.'在念小学<br>';    }
}
function rightstudy($obj) {                  //定义函数，该函数不属于任何类
        if($obj instanceof Students)         //如果该对象是 Students 的实例
            $obj->study();                   //调用该对象的 study()方法
        else  echo '出现错误!<br>';
```

```
}
$s1=new dxs('小新','男');                    //创建 dxs类的对象$s1
rightstudy($s1);
$s2=new xxs('小花','女');                    //创建 xxs类的对象$s2
rightstudy($s2);
$s3=new Students('小文','女');               //创建 Students类的对象$s3
rightstudy($s3);
?>
```

运行程序,输出结果为:

```
小新在读大学
小花在念小学
我在上学
```

说明:程序通过继承 Students 类创建了两个子类:dxs 和 xxs。在两个子类及父类中都定义了 study()方法。通过 instanceof 检测对象类型,这样无论增加多少种 Students 类的子类,都能调用到正确的方法,并且不需要对 rightstudy()函数进行修改。

多态使我们将编程的重点放在接口和父类上,而不必考虑对象具体属于哪个类的问题。

虽然不使用多态,而使用条件判断语句判断参数的个数或类型也能使调用函数时自动调用相应的函数,但那样就不得不在函数中多写很多条件语句来判断,并且使不同的功能都集中到一个函数中了。

习 题 8

一、练习题

1. 如果函数带有多个参数,则参数之间必须用以下哪个符号分开?(　　)

   A. ,　　B. :　　C. &　　D. ;

2. 如果要从函数返回值,必须使用下列哪个关键词?(　　)

   A. continue　　B. break　　C. exit　　D. return

3. 下列关于函数的说法,哪一项是错误的?(　　)

   A. 函数具有重复使用性

   B. 函数名的命名规则和变量命名规则相同,必须以 $ 作为函数名的开头

   C. 函数可以没有输入和输出

   D. 如果把函数定义写在条件语句中,那么必须当条件表达式成立时,才能调用该函数

4. 如果要在函数内定义函数外也可访问的变量,必须使用下列哪个关键词?(　　)

   A. public　　B. var　　C. static　　D. global

5. 如果想保留函数内局部变量的值,必须使用下列哪个关键词?(　　)

   A. private　　B. var　　C. static　　D. global

6. 下列哪个函数可用来取得四舍五入的值？（　　）

A. ceil　　B. floor　　C. round　　D. abs

7. 下列哪个函数可以用来取得次方值？（　　）

A. sqrt　　B. pow　　C. exp　　D. rand

8. 下列哪个函数可以用来取得当前的时间信息？（　　）

A. getdate　　B. gettime　　C. mktime　　D. time

9. 下列哪个函数可以将字符串逆序排列？（　　）

A. chr　　B. ord　　C. strstr　　D. strrev

10. 下列哪个函数可以将数组中各个元素连接成字符串？（　　）

A. implode　　B. explode　　C. str_repeat　　D. str_pad

11. 下列哪个函数可以将换行符转换成 HTML 换行标记？（　　）

A. nl2br　　B. substr　　C. strcmp　　D. strlen

12. 下列哪个运算符可以用来访问对象的成员？（　　）

A. ::　　B. =>　　C. ->　　D. .

13. 下列哪个运算符可以直接访问类内的方法或常量，而无须创建对象？（　　）

A. ::　　B. =>　　C. ->　　D. .

14. 下列哪个语句可以在子类调用父类的构造函数？（　　）

A. base::__construct()　　B. this::construct()

C. parent::__destruct()　　D. parent::__construct()

15. 关于构造函数的说法，下列哪一项是错误的？（　　）

A. 使用 new 创建对象时会自动运行构造函数

B. 名称只能为__construct

C. 子类会继承父类的构造函数

D. 不可以有参数

16. 如果一个对象的实例要调用该对象自身的方法函数 mymeth，则应使用（　　）。

A. $self->mymeth()　　B. $this->mymeth()

C. $current->mymeth()　　D. $this::mymeth()

17. 如果类中的成员声明时没有使用限定字符，则成员属性的默认值是（　　）。

A. private　　B. protected　　C. public　　D. final

18. 在类中定义的析构方法是在什么时候被调用的？（　　）

A. 类创建时　　B. 创建对象时　　C. 删除对象时　　D. 不会自动调用

19. PHP 中调用类文件中的 this 表示（　　）。

A. 用本类生成的对象变量　　B. 本页面

C. 本方法　　D. 本变量

20. 下列关于类的说法，哪项是错误的？（　　）

A. 父类的构造函数与析构函数不会被自动调用

B. 成员变量需要用 public protected private 修饰，在定义变量时不再需要 var 关键字

C. 父类中定义的静态成员,不可以在子类中直接调用

D. 包含抽象方法的类必须为抽象类,抽象类不能被实例化

21. 若要显示“xxxx 年 xx 月 xx 日 星期 x xx:xx:xx”,应设置 date()函数的参数为________。

22. substr('abcdef',1,3)的返回值是________,substr('abcdef',-2)的返回值是________。

23. 如果字符串 $a="test",$b="es",对 $a 进行处理得到 $b 的方法是________。

24. 函数 strpos("xxPPppXXpx","pp")的返回值是________。

25. 要截取中文字符串,并且不产生乱码,应使用的函数是________。

26. 变量 $this 指的是对象本身,对不对?

27. PHP 允许父类有多个子类,也允许子类有多个父类,对不对?

二、编程题

1. 用 PHP 输出前一天的时间,要求格式为 2006-5-10 22:10:11。

2. 编写一个实现字符串反转的函数。

3. 编写一个函数,使用字符串处理函数获得文件的扩展名,如输入 ab.jpg,输出 jpg。

4. 编写一个函数,输入是一个小于 8 位的任意位数的整数,输出是这个整数各个位上的数。要求分别用两种方式实现:

(1) 直接在函数内部用 echo 语句输出,函数没有返回值;

(2) 用字符串处理函数截取该整数各位上的数。函数的返回值是一个数组,数组中各元素保存了各个位上的数。

5. 编写一个可计算某整数四次方的函数,该函数的输入是一个整数,输出是该数的四次方。调用该函数计算 16 的四次方,并输出结果。

6. 编写一个用来判断某整数是否是质数的函数,该函数的输入是一个整数,如果该整数是质数,就返回 true,否则返回 false,然后调用这个函数输出 2~100 之间所有的质数。

7. 任意输入一个整数,使用函数的方法判断该数是否为偶数。

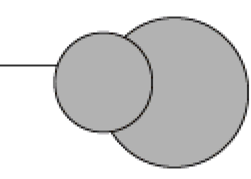
8. 编写一个函数,实现以下功能,将字符串"cute_boy"转换成"CuteBoy",将"how_are_you"转换成"HowAreYou"。

9. 编写一个函数,输入是 5 个分数,输出是去掉一个最高分和一个最低分后的平均分。

10. 将 8.1.1 节中的例 8.1 改写成函数,即输入是待过滤的字符串和非法字符集,输出是过滤后的字符串,并调用该函数实现例 8.1 的功能。

11. 编写函数,计算两个文件的相对路径(例如,“$a='/a/b/c/d/e.php'; $b='/a/b/12/34/c.php';”,则计算出 $b 相对于 $a 的相对路径应该是../../c/d)。

第9章



Web交互编程

Web 应用程序的基本功能就是与用户进行交互,获取并处理用户提交的数据。用户提交数据的方法有:

(1) 通过表单提交,如用户注册、用户登录、留言等都是通过表单提交信息。

(2) 使用网址中的 URL 参数发送数据给服务器。这些数据都是以 HTTP 请求的方式发送。Web 服务器必须能够获取用户通过浏览器发送来的数据,才能与用户进行交互。

9.1 接收浏览器数据

浏览器可以通过表单或 URL 字符串向服务器发送数据,这些数据称为 HTTP 请求信息。PHP 提供了很多预定义的超全局变量,如表 9-1 所示,用来获取 HTTP 请求信息,这些超全局变量的数据类型均为数组。

表 9-1 PHP 的超全局变量及功能

超全局变量	功 能
$_POST	获取客户端以 POST 方式发送的 HTTP 请求信息
$_GET	获取客户端以 GET 方式发送的 HTTP 请求信息
$_REQUEST	包含了 $_GET、$_POST 和 $_COOKIE 三类数组中的信息
$_SERVER	获取 HTTP 请求中的环境变量信息
$_SESSION	存储和读取为单个用户保存的信息
$_COOKIE	读取用户的 Cookie 信息
$_FILE	获取通过 POST 方式上传文件时的相关信息,为多维数组
$_ENV	获取服务器名称或系统 shell 等与服务器相关的信息

说明:所谓超全局变量,表示该变量在一个文件的所有区域中都可使用,包括自定义函数内部。

9.1.1 使用 $_POST[]获取表单数据

用户在表单中输入数据后,可以单击“提交”按钮,将数据提交给服务器,由在服务器

端工作的 PHP 程序接收和处理这些表单数据。

表单提交数据的方式分为 GET 和 POST 两种，在定义表单时，将 method 属性设置为 GET 或不设置时，都会采用 GET 方式提交；将 method 属性设置为 POST 时，则会采用 POST 方式提交。

使用 GET 方式提交数据时，表单数据将通过 URL 参数的形式发送给服务器，而使用 POST 方式时，数据不会出现在 URL 参数中。

在 PHP 中，$_POST 数组用来获取使用 POST 方式提交的表单数据，语法如下：

```
变量名=$_POST["表单元素 name 属性值"]
```

表示将获取的某个表单元素值（如一个文本框中的内容）保存到一个服务器端变量中。例如，$user=$_POST["user"]，其中，$user 是自定义的一个变量名称，而后面的 user 则是一个表单元素的名称（name 属性值），两者不是一回事。

在实际应用中，表单的 HTML 代码和获取表单数据的 PHP 程序既可以分别写在两个文件中，也可以写在同一个文件中。下面分别来讲述。

1. 使用两个网页文件

下面的例子用来获取用户登录时输入的用户名和密码。它使用了两张网页，其中 9-1.php 用来显示表单，是一个纯 HTML 页面，9-2.php 用来接收并输出获取的表单数据。

```
----------------------清单 9-1.php----------------------------
<html><body>
<form method="post" action="9-2.php">
  用户名：<input type="text" name="userName" size="12">
  密码：<input type="text" name="PS" size="10">
  <input type="submit" value="登录">
</form>
</body></html>
----------------------清单 9-2.php----------------------------
<html><body>
<?  $userName=$_POST["userName"];
    $PS=$_POST["PS"];
    echo "您输入的用户名是:".$userName;
    echo "<br>您输入的密码是:".$PS;
?>
</body></html>
```

9-1.php 的运行结果如图 9-1 所示，单击“登录”按钮，就会将表单数据提交给 9-2.php，9-2.php 接收并显示数据，如图 9-2 所示。

请注意表单代码中有几个关键属性与接收表单数据的程序密切关联，如图 9-3 所示。

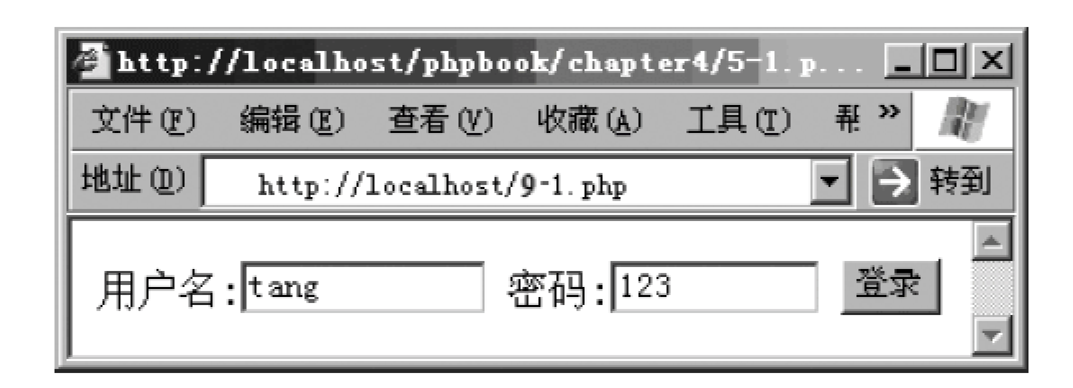


图 9-1　9-1.php 的运行结果



图 9-2　9-2.php 的运行结果

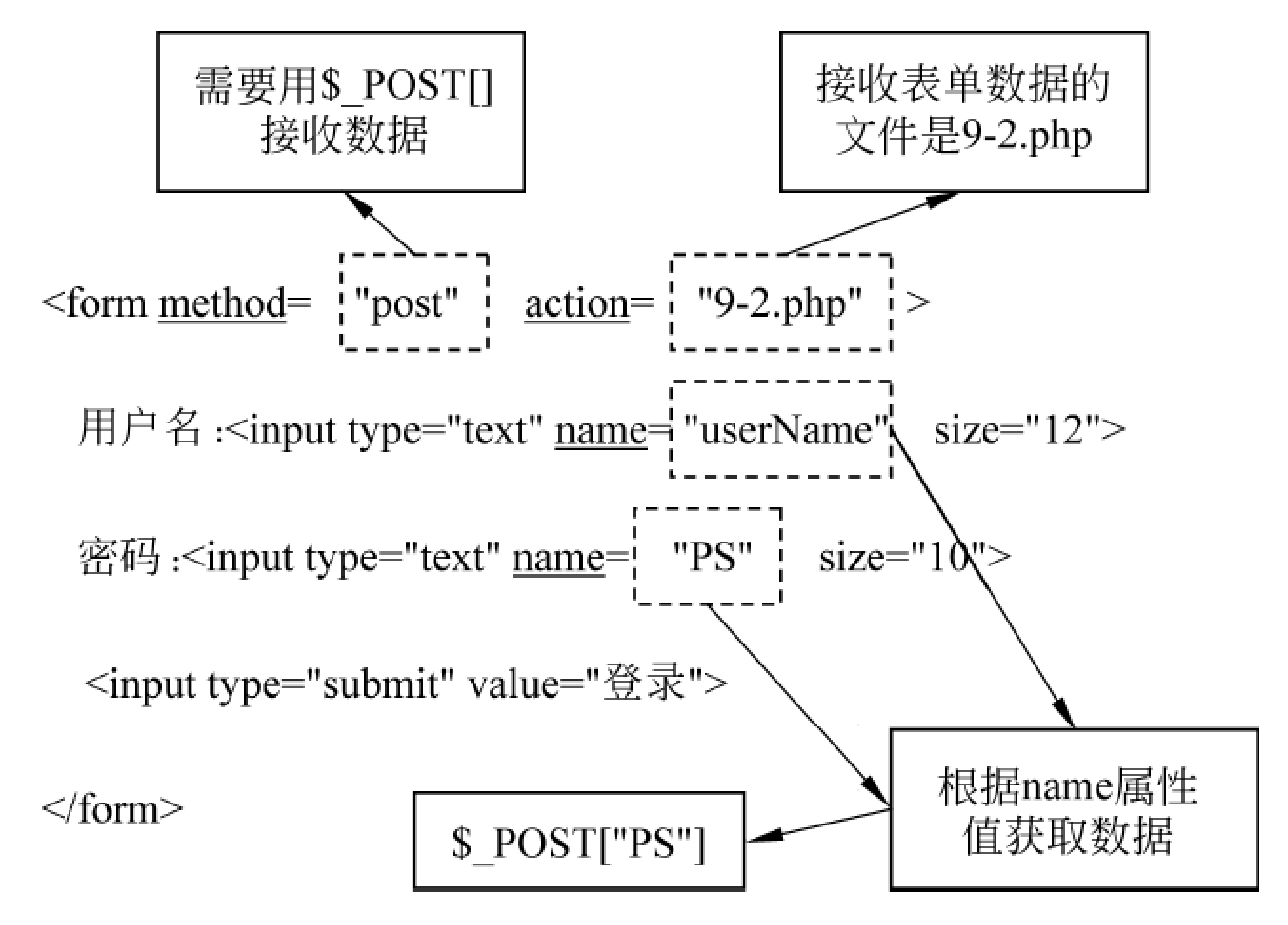


图 9-3　表单代码中与接收数据密切相关的属性

说明：

(1) 在 9-1.php 中，<form>标记的 method 属性值为 post，表示该表单提交数据时以 POST 方式提交。如果将其改为 get，那么将以 GET 方式提交，此时必须用 $_GET[]才能获取表单数据。

(2) <form>标记的 action 属性用来指定接收表单数据的文件，它的属性值可以是相对 URL 或绝对 URL。这里因为两个文件在同一个文件夹下，直接写文件名即可。由于只有服务器端程序才能接收表单数据，因此该文件中必须含有 PHP 代码。

(3) 9-1.php 中有 3 个表单元素：2 个文本框和 1 个提交按钮，通过表单元素的 name 属性值可以获取该表单元素中输入的内容(即 value 属性值)，其中 $_POST["userName"]会返回第一个文本框中输入的值(文本框会将框中的内容作为其 value 属性值)，$_POST["PS"]会返回第二个文本框中输入的值。而由于没有对提交按钮设置 name 属性，因此它的 value 值不会发送给服务器。

(4) 在 9-2.php 中，也可以不将 $_POST 的值赋给变量而是直接使用，例如，echo "您输入的用户名是:". $_POST["userName"]。但为了方便引用，也为了能对获取的值先进行一些检验处理(如过滤非法字符或空格，本例省略)，最好先用一个变量引用它。

2. 使用一个网页文件

上述示例中的两个文件可以合并为一个文件。也就是说，网页可以将表单中的信息提交给自身。这样做的好处是减少了网站内网页文件数量。

实现的方法是：设置<form>标记的 action=""或 action="自身文件名"，然后将表单代码和 PHP 代码写在同一个文件中，并判断只有在用户提交了表单后才执行 PHP 代码，代码如下，运行效果如图 9-4 所示。

------------------------清单 9-3.php------------------------

```
<html><body>
<form method="post" action="">
  用户名 :<input type="text" name="userName"   size="12">
  密码 : <input type="text" name="PS"   size="10">
  <input type="submit" name="denglu" value="登录">
</form>
<?
if(isset($_POST['denglu'])) {      //判断用户是否提交了表单(即单击了提交按钮)
     $userName=$_POST["userName"];
     $PS=$_POST["PS"];
     echo "您输入的用户名是:".$userName;
     echo "<br>您输入的密码是:".$PS;      }      ?>
</body></html>
```

说明：

(1) 本例中，将 9-2.php 中的 PHP 代码全部放在一个条件语句 if(isset($_POST['denglu'])){…}中。它表示，如果用户单击了“登录”按钮($_POST['denglu']变量就会存在)，才执行该条件语句中的内容：获取表单数据并输出。因此，当用户刚打开页面时，还未单击提交按钮，就不会执行条件语句中的内容，只会显示表单。



图 9-4 9-3.php 的执行结果

(2) 提交表单就会刷新一次网页，而刷新页面就会将页面中的所有代码重新执行一次。因此用户单击提交按钮后，9-3.php 的代码会从头到尾重新执行一遍。

想一想：当用户输入信息后，9-3.php 同时显示了表单界面和获取的信息，如果只希望输出获取信息，而不再显示表单，即和 9-2.php 执行效果一模一样，该怎么修改 9-3.php 呢？

3. 获取复杂一点的表单页面

下面是一个获取用户注册信息的例子，其中 9-4.php 用来显示表单，9-5.php 用来获取表单数据。请仔细体会获取单选框、复选框、下拉框和多行文本域等表单元素中内容的方法。

------------------------清单 9-4.php------------------------

```
<html><body>
     <h1 align="center">新用户注册</h1>
     <form method="Post" action="9-5.php">
```

```
    姓名：<input type="text" name="name"><br>
    性别：<input type="radio" name="Sex" value="1" checked="checked">男
          <input type="radio" name="Sex" value="0">女<br>
    爱好：<input type="checkbox" name="hobby[]" value="太极拳">太极拳
              <input type="checkbox" name="hobby[]" value="音乐">音乐
              <input type="checkbox" name="hobby[]" value="旅游">旅游<br>
    职业:<select name="career">
                <option value="教育业">教育业</option>
                <option value="医疗业">医疗业</option>
                <option value="其他">其他</option>
        </select><br>
    个性签名:<textarea name="intro" rows="2" cols="20"></textarea><br>
        <input type="submit" value="  提 交  ">
    </form>
</body></html>
----------------------清单 9-5.php------------------------
<html><body>
    <h3 align="center">
    <?  $name=$_POST["name"];              //获取各个表单元素的值
    $Sex=$_POST["Sex"];
    $hobby=$_POST["hobby"];
    $career=$_POST["career"];
    $intro=$_POST["intro"];
    $hobbynum=count($hobby);
    echo  "尊敬的".$name;                   //输出各个表单元素的值
    if($Sex=="1") echo "先生</h3>";         //根据单选框的值输出先生或女士
    if($Sex=="0") echo "女士</h3>";
    echo "<p>您选择了".$hobbynum."项爱好:</p>";
    for($i=0;$i<$hobbynum;$i++)            //通过循环输出复选框的值
      echo $hobby[$i].' ';
    echo "<br>您的职业：" .$career;
    echo "<br>您的个性签名：" .$intro;
    //var_dump($_POST);                     //获取所有表单元素的值
?>
    <p><a href="JavaScript:history.go(-1)">返回修改</a></p>
</body></html>
```

程序 9-4.php 的初始运行效果如图 9-5 所示，单击“提交”按钮后效果如图 9-6 所示。

说明：

(1) 对于单选框，两个单选框的 name 属性值一样，就表示这是一组，只能选中一个。

(2) 对于复选框，三个复选框的 name 属性值相同，也表示是一组，但复选框可以选择多个，如果选中多个，则多个复选框的值将保存在一个数组中，可以用循环语句输出所有选中复选框的值。

(3) 总的来说，表单元素可分为两类：



图 9-5 9-4. php 的初始运行结果

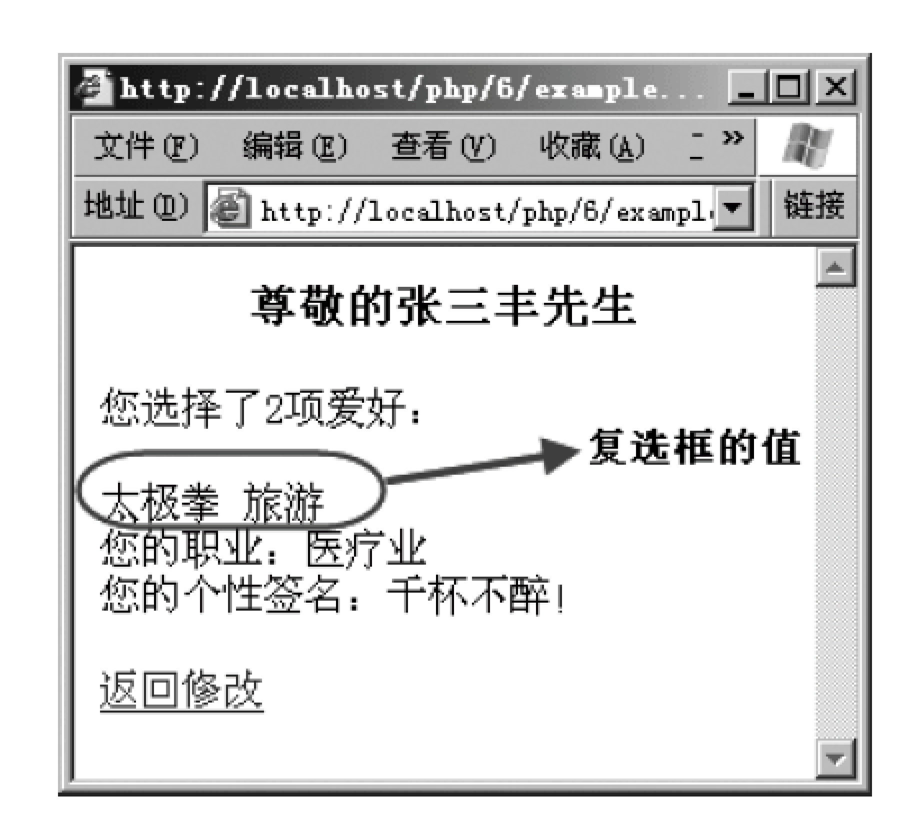


图 9-6 9-4. php 单击提交按钮后

① 对于文本框、密码框、多行文本框这些需要用户输入内容的，$ _POST[]获取的就是用户输入的内容；

② 对于单选框、复选框、下拉列表框、隐藏域这些无须用户输入内容的，$ _POST[]获取的就是选中项的 value 值。

因此对第②类表单元素必须设置 value 属性值。

(4) 当多个复选框属于同一组时，对其 name 属性值命名，一定要命名成数组的形式，如 name=" hobby[]"。这样 $ _POST["hobby"]返回的结果才会是一个数组，可以利用数组名加索引值来取得其中某个复选框的 value 值，例如 $ _POST["hobby"][1]的值为“旅游”。还可利用 count()方法获得该数组中的元素总数，例如 count($ hobby)。

4. 对 $ _POST 数组的深入认识

实际上，$ _POST 是一个数组，它保存了接收到的所有的表单元素值，而 $ _POST ["Sex"]是一个数组元素。我们可以在 9-5. php 中添加一行代码：

```
var_dump($_POST);
```

则输出结果为：

```
array(5) { ["name"]=> string(6) "张三丰" ["Sex"]=> string(1) "1" ["hobby"]=> array(2) { [0]=> string(6) "太极拳" [1]=> string(4) "旅游" } ["career"]=> string(6) "医疗业" ["intro"]=> string(10) "千杯不醉!" }
```

可见，在该数组中，数组元素的索引值为表单元素的 name 属性值，数组元素的值为表单元素的 value 属性值。因此可以用数组元素如 $ _POST["name 属性值"]获取到对应表单元素的 value 属性值。

$ _POST 数组元素的索引一定要加引号，如 $ _POST["Sex"]，不加虽然不会出错，但运行效率会大大降低，原因请参见 7. 5. 1 节中“创建数组的注意事项”。

9. 1. 2 使用 $ _GET[]获取表单数据

如果表单是以 GET 方式提交的，即 method 属性值为 GET(或没有设置)，则必须用

$ _GET[]数组获取表单中的数据。读者可将 9-1.php 中的 method="post"修改为 method="get",将 9-2.php 中所有的 $ _POST 修改为 $ _GET,就会发现仍然能正常获取表单中数据,运行效果类似于图 9-1 和图 9-2。

但 GET 方式与 POST 方式提交是有区别的,GET 方式会将表单中的数据以 URL 字符串的形式发送给服务器,例如,当 9-1.php 以 GET 方式提交后,浏览器地址栏会显示:

```
http://localhost/9-1.php?userName=tang&PS=123
```

而以 POST 方式提交的话,浏览器地址栏只会显示:

```
http://localhost/9-1.php
```

可见,POST 方式提交表单比 GET 方式提交表单更安全,不会泄露机密数据。并且,POST 方式发送数据时对字节数没有限制。

9.1.3 使用 $ _GET[]获取 URL 字符串信息

URL 字符串又称为查询字符串,这些字符串信息总是以 GET 方式提交给服务器,可以使用 $ _GET[]方法获取 URL 字符串信息。

1. 什么是 URL 字符串

有些网页的 URL 后面会接一些以"?"号开头的字符串,这称为 URL 字符串。例如:

```
http://ec.hynu.cn/otype.php?owen1=近期工作&page=2
```

其中,"?owen1=近期工作&page=2"就是一个 URL 字符串,它包含 2 个 URL 变量(owen1 和 page),而"近期工作"和"2"分别是这两个 URL 变量的值,URL 变量和值之间用"="号连接,多个 URL 变量之间用"&"连接。

URL 字符串会连同 URL 信息一起作为 HTTP 请求数据提交给服务器端的相应文件,例如上面的 URL 字符串信息将提交给 otype.php。利用 $ _GET[]可以获取 URL 字符串中变量的值。例如,在 otype.php 中编写如下代码,就能获取到这些查询变量的值了。

```
<?
    $owen=$_GET["owen1"];        //获取变量 owen1 的值,返回"近期工作"
    $page=$_GET["page"];         //获取变量 page 的值,返回"2"
?>
```

2. 设置 URL 字符串的方法

当网页通过超链接或其他方式从一张网页跳转到另一张网页时,往往需要在跳转的同时把一些数据传递到第二张网页中。我们可以把这些数据作为 URL 字符串附在超链接的 URL 后,就可以在第二张网页中使用 $ _GET[]获取 URL 变量的值。例如:

```
<a href="search.php?key=Web标准&pageNo=5">查询结果第 5 页</a>
```

则在 search.php 中就可用 $_GET[]获取第一张网页传递来的 URL 变量的值。这是通过超链接设置 URL 字符串,第二种方法是在<form>标记的 action 属性中设置。下面分别举例介绍。

例 9.1 在超链接中设置 URL 字符串并获取,运行结果如图 9-7 和图 9-8 所示。

```
----------------------清单 9-6.php------------------------
<html><body>
<ul>
  <li><a href="9-9.php?id=1">《电子商务安全》震撼上市</a></li>
  <li><a href="9-9.php?id=2">ASP动态网页设计与 Ajax技术</a></li>
  <li><a href="9-9.php?id=3">基于 Web标准的网页设计与…</a></li>
</ul>
</body></html>
----------------------清单 9-7.php------------------------
<?  $id=intval($_GET["id"]);            //获取 URL字符串中变量 id的值并转为整型
    if($id==1)
        echo "<p>这是第一条新闻</p>";
    elseif($id==2)
        echo "<p>这是第二条新闻</p>";
    elseif($id==3)
        echo "<p>这是第三条新闻</p>";
    else   echo "<p>参数非法</p>";   ?>
```



图 9-7 单击 9-6.php 中第二个超链接



图 9-8 9-7.php 的运行结果

说明：

(1) 9-6.php 中所有链接都是链接到同一网页,只是设置了不同的 URL 变量值,可以使 9-7.php 根据不同链接传来的不同 id 值显示对应的网页内容,实现了动态新闻网页效果。

(2) URL 变量中的数据都是字符串类型的值,因此如果要对数值进行判断,最好先转换为数值型,这样可防止非法用户手工在 URL 后注入非法参数。

例 9.2 在<form>标记的 action 属性中设置 URL 字符串。运行结果如图 9-9 所示。

```
----------------------清单 9-8.php------------------------
<html><body>
<?  $flag=$_GET["flag"];        //获取 url变量 flag的值
```

```
if($flag=='1')
    echo '欢迎 '.$_POST['user'].' 光临!';
else                                    //没有按提交按钮时
    echo '<form method="post" action="?flag=1">
        姓名 :<input name="user" type="text" size="15" />
        <input type="submit" value="提交" />
        </form>';       ?>
</body></html>
```



图 9-9　9-8.php 的运行结果

说明：在<form>标记中，“action="?flag=1"”省略了文件名，表示将表单提交给自身，设置查询字符串 flag=1 用来判断用户是否单击了“提交”按钮，一旦单击了则 URL 地址后面会增加"?flag=1"，因此可据此输出不同的内容。

3. 设置 URL 字符串的方法总结

如果要设置 URL 字符串，以便将 URL 字符串中的信息传递给相应网页。有以下方法：

(1) 在超链接的 href 属性值中的 URL 后添加 URL 字符串；

(2) 在表单的 action 属性值中的 URL 后添加 URL 字符串；

(3) 直接在浏览器地址栏中的网页 URL 后手工输入 URL 字符串。

提示：表单如果设置为 GET 方式提交，那么表单中数据将转换成 URL 字符串发送给服务器。此时若在表单的 action 属性值中也设置 URL 字符串，那么将发生冲突，action 属性值中的 URL 字符串将无效。因此如果在 action 属性值中有 URL 字符串，则表单只能用 POST 方式提交，9-8.php 就是一个例子。

9.1.4　发送 HTTP 请求的基本方法

浏览器向服务器发送 HTTP 请求有两种基本方法：一种方法是在地址栏输入网址并回车，这样将以 GET 方式向服务器发送一个 HTTP 请求；另一种方法是提交表单，如果设置 form 标记的 method 属性为 get，那么表单中的数据将以 GET 方式发送给服务器；如果设置 form 标记的 method 属性为 post，那么数据将以 POST 方式发送。对于 GET 方式的 HTTP 请求，服务器端只有使用 $_GET[]才能获取其中的数据，而对于 POST 方式的 HTTP 请求，服务器端只有使用 $_POST[]才能获取其中的数据，如表 9-2 所示。

表 9-2　浏览器发送请求与服务器获取请求的方法

<table>
<tr><th colspan="2">发送 HTTP 请求</th><th>发送请求的方式</th><th>服务器获取请求的方法</th></tr>
<tr><td colspan="2">输入网址(URL)</td><td rowspan="2">POST 方式</td><td rowspan="2">$_POST[]</td></tr>
<tr><td rowspan="2">提交表单</td><td>method="get"</td></tr>
<tr><td>method="post"</td><td>GET 方式</td><td>$_GET[]</td></tr>
</table>

一个 HTTP 请求实际上是一个数据包，如果以提交表单形式发送 HTTP 请求，则这个数据包中含有表单数据，如果是 GET 方式发送的请求，则包含了 URL 字符串中的数据。

以 9-1.php 和 9-2.php 的执行过程为例。我们可以把浏览器发送的 HTTP 请求数据包想象成一辆卡车，它装载了用户在表单中填写的信息。当单击“提交”按钮后，就会发送 HTTP 请求给服务器。这就好比这辆卡车载着货物(表单中的信息)从浏览器行驶到了服务器。服务器此时可以使用 $_POST[]卸下卡车上的货物，保存到服务器端的变量中。整个过程如图 9-10 所示。

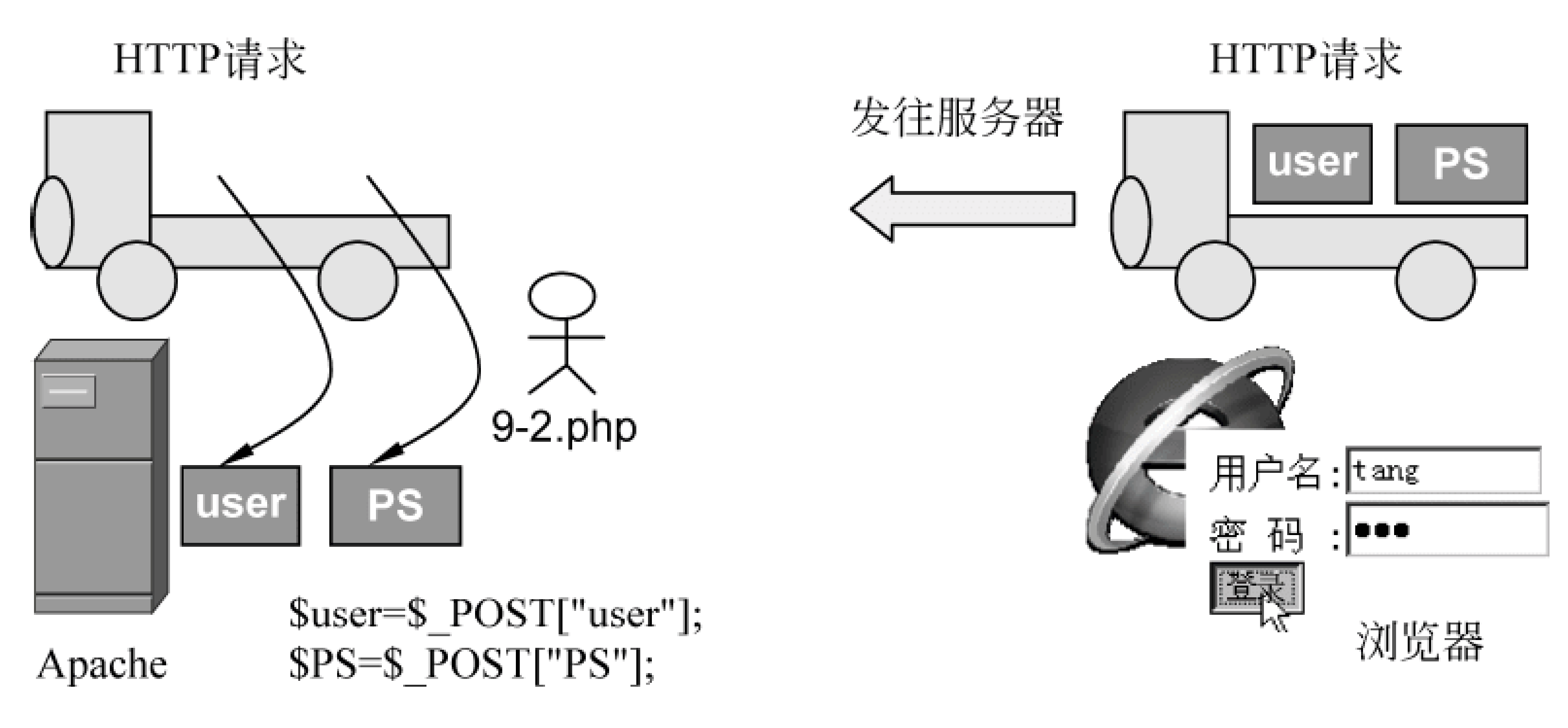


图 9-10　浏览器发送 HTTP 请求和服务器获取 HTTP 请求内容的示意图

提示：PHP 还提供了 $_REQUEST[]数组，它包含了 $_GET、$_POST 和 $_COOKIE 数组信息。因此它可以获取 GET 或 POST 两种方式提交的数据，以及 Cookie 数据。所以，PHP 程序中的 $_GET 或 $_POST 都可换成 $_REQUEST。

9.1.5　使用 $_SERVER[]获取环境变量信息

实际上，浏览器发送的 HTTP 请求数据包中还包含了客户端的 IP 地址、请求文件的 URL 等环境变量信息。服务器在接收到这个请求时也会给出服务器端 IP 地址等环境变量信息，利用 $_SERVER[]数组可以方便地获取到这些信息。例如获取浏览者 IP 和来路的程序如下：

```
--------------------清单 9-9.php--------------------------------
<?  $IP=$_SERVER['REMOTE_ADDR'];      //获取用户 IP地址
    $From=$_SERVER['HTTP_REFERER'];//获取用户来路
    echo "您的 IP地址是：".$IP;
```

```
    echo '<br>您是单击'.$From.'页面中的链接进来的'
?>
```

在程序中,REMOTE_ADDR 就是一个环境变量名,表示客户端的 IP 地址。而 HTTP_REFERER 可以获取用户是从哪个网页进入当前网页的。例如,在 9-6.php 中做一个到 9-9.php 的超链接,单击该超链接进入 9-9.php,则 9-9.php 的运行结果如下:

```
您的 IP地址是:127.0.0.1
您是单击 http://localhost/6/9-8.php 页面中的链接进来的
```

通过 HTTP_REFERER 获取用户来路,可以知道自己的网页被哪些网站收录或反链,例如,如果用户是单击"百度"上的搜索结果打开当前网页的,那么获取来路就会返回百度搜索页的 URL。

其他比较有用的环境变量如表 9-3 所示。

表 9-3 常用的环境变量

环境变量名	功 能 说 明
QUERY_STRING	URL 字符串信息
SCRIPT_NAME	当前文件相对于网站目录的路径和文件名
SCRIPT_FILENAME	当前文件在硬盘中的路径和文件名
PHP_SELF	当前正在执行脚本的文件名
DOCUMENT_ROOT	当前网站根目录
SERVER_SOFTWARE	Web 服务器软件的名称,如 IIS/5.1
SERVER_PORT	服务器端的端口号
SERVER_NAME	服务器主机的名称
SERVER_SIGNATURE	包含服务器版本和虚拟主机名的字符串
REQUEST_METHOD	HTTP 请求的方式,如 GET、POST
REQUEST_TIME	HTTP 请求开始时的时间戳

9.2 发送数据给浏览器

Web 应用程序的基本功能包括两方面:一是接收并处理浏览器发送的 HTTP 请求数据,二是根据 HTTP 请求做出 HTTP 响应,这包括输出信息给浏览器,使浏览器重定向到其他页面等。

9.2.1 使用 echo 方法输出信息

在 PHP 中,echo()是最常用的方法,它用来将服务器端的数据发送给浏览器。所发送的信息可以是字符串常量、变量、HTML 代码、JavaScript 代码等所有浏览器能解释的

代码。下面是使用 echo 方法输出信息的示例：

```
<?    echo "欢迎您：";                              //输出字符串常量
    $i='小花';
    echo $i;                                        //输出变量
    echo '<p>欢迎您：'.$i .'</p>';                  //输出字符串常量和变量的混合体
    echo "<a href='9-4.php'>新用户注册</a>";        //输出 HTML 代码
    echo "<script>alert('留言修改成功');location.href='9-8.php';
</script>";
    echo "<br>欢迎您：",$i;                         //输出多个字符串   ?>
```

说明：

（1）echo 后可接字符串常量或变量。回顾一下字符串常量和变量的写法：两边加引号表示字符串常量，不加引号表示的是变量。如果要输出的内容既有字符串常量又有变量，则它们之间要用连接符（.）连接或使用双引号字符串包含变量。

（2）HTML 代码本质上也是一段字符串，echo 可将它作为字符串常量输出，因此 HTML 代码两边要加引号。

（3）echo 方法可以加括号，也可以不加括号，如“echo "欢迎"”也可写成“echo("欢迎")”。

（4）echo 还有一种省略的写法，即<? =… ? >（“<?”和“=”之间不能有空格），例如：

```
<?="欢迎您：";    ?>
<?='<p>欢迎您：'.$i .'</p>';    ?>
```

这种方法虽然简便，但它的两端必须要有“<?”和“? >”，导致在它前面和后面的 PHP 代码也必须用“? >”和“<?”进行封闭。如果它的前面和后面都是 HTML 代码，则用这种方式比较方便。如果它的前后都是 PHP 代码，则使用 echo 方法更清晰。

提示： 在 PHP 中，还有 print、print_r、var_dump 这些语句也能用来输出信息。

echo 和 print 功能几乎完全相同，唯一的区别是：使用 echo 可以同时输出多个字符串，多个字符串之间用逗号隔开即可，而 print 一次只能输出一个字符串。

print_r()用于输出整个数组，var_dump()用于输出变量的数据类型和值，是调试程序的好帮手。这两个方法后面的括号都不能省略。

9.2.2 使用 header()函数重定向网页

在 HTML 中，可以使用超链接引导用户至其他页面，但必须要用户单击超链接才行。可是有时需要自动引导（也称为重定向）用户至另一页面（如用户注册成功后就自动跳转到登录页面）。或者根据程序来动态判断将用户引导到哪一页面。

1. 重定向网页

在 PHP 中，使用 header()函数可以重定向网页，例如：

```
<?
header("location:http://www.baidu.com");      //重定向到绝对 URL
header("location:9-8.php");                    //重定向到相对 URL
header("location:?flag=1");                    //重定向到本页,并增加 URL字符串
$url='9-1.php';
header("location:$url");                       //重定向到变量表示的网址
?>
```

注意:

(1) location 和":"号之间不能有空格,否则会出错;

(2) 在 header 语句之前,服务器不能向客户端发送任何数据,比如不能有 echo 语句也不能有 HTML 代码(包括空行);

(3) 如果 PHP 代码中有多条重定向语句,则会重定向到最后一条语句中的 URL,表明 PHP 在执行重定向语句后,仍然会继续执行后面的语句。如果希望执行完重定向语句后立即停止脚本执行,应在 header 语句后使用 exit()或 die()方法退出。

header 函数的功能和 JavaScript 脚本中 location. href 的功能有些相似,如"header("location:9-8. php");"又可使用"echo "<script> location. href='9-8. php';</script>";"来实现。不过 header()函数要求在重定向之前不允许服务器向浏览器输出任何内容,因此使用该方法要么确保先用 ob_start()打开服务器缓冲区,使所有的内容先输出到缓存中,还没有输出到浏览器;要么确保在 header()语句之前没有任何内容输出到页面。因此下面的写法是错误的:

```
<html><body>      <!--错误,header语句前不能输出内容到浏览器-->
<?    header("location:9-8.php ");   ?>
</body></html>
```

2. header()函数的其他功能

实际上,header()函数的功能是向浏览器传送一个 HTTP 响应头信息,语法如下:

```
void header(string message [, bool replace [, int http_response_code]])
```

其中,message 参数用来设置响应头信息,其格式为"header_name: header_value"。

浏览器收到这些响应头信息后,会做出适当的反应。如收到"Location: URL"响应头后,浏览器就会将页面重定向到 URL 指定的页面。header()函数的其他功能如下:

1) 文件延迟转向

使用 Refresh 响应头,可以使页面延迟 N 秒后,重定向到指定的 URL 页面。例如:

```
header('Refresh:3; url=http://ec.hynu.cn');      //3秒后转到 ec.hynu.cn
```

2) 禁用浏览器缓存

为了让用户每次都能从服务器上获取最新的网页,而不是浏览器缓存中的网页,可以使用下列标头禁用浏览器缓存。

```
header('Expires: Mon,26 Jul 1997 05:00:00 GMT');      //设置过期时间为过去某一天
```

```
header('Last-Modified:' . gmdate('D, d M Y H:i:s') . 'GMT');
header('Cache-Control: no-store, no-cache, must-revalidate');
header('Pragma: no-cache');
```

3）强制下载文件

简单文件下载只需要使用超链接标记<a>，将 href 属性值指定为下载的文件即可。这种下载方式，只能处理一些浏览器不能打开的文件（如 rar 文件），但如果要下载的文件后缀名是. html 的网页文件或图片文件等，使用这种链接方式并不会提示下载，而是将文件内容直接输出到浏览器。为此，可使用 header 函数向浏览器发送必要的头信息，以通知浏览器进行下载文件的处理。强制下载文件的示例代码如下：

```
<?  $filename="test.gif";                      //指定文件名
    header('Content-Type: image/gif');         //指定下载文件类型
    header('Content-Disposition: attachment; filename="'.$filename.'"');     //下载文件的描述
    header('Content-Length: '.filesize($filename));                          //下载文件的大小
    readfile($filename);        //将文件内容读取出来并直接输出,以便下载
?>
```

这样，在当前目录下放一个图片文件 test. gif，运行程序，就会提示下载 test. gif 文件。

9.2.3 操作缓冲区

所谓缓冲区，是指服务器内存中的一块区域。在没有开启缓冲区时，执行文件输出的内容都是直接输出到浏览器。开启缓冲区后，执行文件输出的内容会先存入缓冲区，直到脚本执行完毕（或遇到一些缓冲区操作指令），再将缓冲区中的内容发送给浏览器。

PHP 提供了很多操作缓冲区的函数（见表 9-4），这些函数名中都有“ob”，ob 是“Output Control”的缩写，表示在服务器端先存储有关输出，等待适当的时机再输出。下面分别介绍。

表 9-4 PHP 缓冲区操作函数

函 数 名	功 能
ob_start	打开输出缓冲区
ob_get_contents	返回内部缓冲区的内容
ob_get_clean	返回内部缓冲区的内容，并关闭缓冲区
ob_get_flush	返回内部缓冲区的内容，并关闭缓冲区，再将缓冲区的内容立刻输出到客户端
ob_get_length	返回内部缓冲区的内容长度
ob_clean	删除内部缓冲区的内容，但不关闭缓冲区
ob_flush	立刻输出内部缓冲区的内容，但不关闭缓冲区

续表

函　数　名	功　　能
flush	刷新输出缓冲,将 ob_flush 输出的内容,以及不在 PHP 缓冲区的内容,全部输出至浏览器
ob_end_clean	删除内部缓冲区的内容,并关闭缓冲区
ob_end_flush	立刻输出内部缓冲区的内容,并不关闭缓冲区

提示:缓冲区函数名中,end 表示关闭缓冲区;clean 表示删除缓冲区中的内容;flush 表示发送缓冲区中的内容到浏览器;get 表示缓冲区中的内容将作为函数的返回值返回。

1. 使用 ob_start()打开缓冲区

ob_start()用来打开缓冲区,下面的程序用来演示打开和关闭缓冲区时的差异。

```
<?  ob_start();                    //删除该语句再试试
for($i=1;$i<20;$i++){
        for($j=1;$j<600000;$j++);      //空循环语句,用于延迟
        echo $i." ";
}   ?>
```

当程序中有 ob_start()语句时,缓冲区打开,程序会将输出的内容先存储到缓冲区,待程序执行完后,再将缓冲区中的内容一起输出到浏览器。因此运行程序后,会先延迟一段时间,然后所有数字一起显示出来。

将 ob_start()语句删除后,缓冲区关闭,程序每次执行到输出语句就会立即输出,因此运行程序不会延迟,数字会一个一个地显示出来。

2. ob_flush 和 ob_ clean()方法

当缓冲区打开后,ob_flush()方法可以将缓冲区中的内容立刻输出到客户端,ob_clean()方法用于将当前缓冲区中的内容全部清空。例如:

```
<?  ob_start();
    echo "第一条";
    ob_flush();                      //立刻输出缓冲区中的内容
    echo "第二条";
    ob_ clean();                     //清除缓冲区中的内容
    echo "第三条";
    ob_end_flush();                  //发送缓冲区的内容到浏览器,并且关闭缓冲区
?>
```

程序运行结果为"第一条 第三条"。

由于 ob_flush 会将缓冲区中的内容立刻输出,因此"第一条"会显示在页面上,然后"第二条"又被输出到缓冲区,但接下来 ob_clean()方法清除了缓冲区中的内容,因此"第二条"不会显示;"第三条"不受影响,也会输出到缓冲区再输出到页面。

总结：PHP 会在以下三种情况下将缓冲区中的内容发送给客户端：

(1) 遇到 ob_flush()、ob_end_flush()或 ob_get_flush()函数；

(2) 程序执行完；

(3) 遇到 exit 或 die 函数提前终止程序。

3. ob_get_contents 和 ob_get_length 函数

当缓冲区打开时，ob_get_contents 可以获取缓冲区中的内容，并将内容作为字符串返回。ob_get_length 将返回缓冲区中内容的长度，返回值为整数 int 型。下面是一个例子。

```
<?    ob_start();
    echo "<b>Hello World</b>";          //输出到缓冲区,但不会立即输出到页面
    $len=ob_get_length();               //$len保存了当前缓冲区中内容的长度
    $out=ob_get_contents();             //$out保存了当前缓冲区中的内容
    ob_end_clean();                     //清空并结束缓冲区
    echo $out.'<br>';                   //输出<b>Hello World</b>
    $out=strtolower($out);              //将变量$out中的字符转换为小写
    var_dump($out,$len);                ?>
```

输出结果为(注意第一条 echo 语句中的内容未输出，它被 ob_end_clean 方法清除了)：

```
<b>Hello World</b><br>string(18) "<b>hello world</b>"    int(18)
```

可见，ob_get_contents 函数可以将缓冲区中的内容保存到一个字符串中。

9.3 使用 $_SESSION 设置和读取 Session

网站经常需要记住访问者的一些信息，以便提供更好的服务。例如用户登录以后，网站中的所有页面都能显示用户的登录名，这就需要在整个网站中使用一种“全局变量”保存用户名。但是普通变量的作用域只能在一个网页内，当用户从一张网页跳转到另一张网页时，前一张网页中以变量、常量形式存放的数据就丢失了。为此，引入 Session 的概念，只要把用户的信息存储在 Session 变量中，用户在网站页面之间跳转时，存储在 Session 变量中的信息不会丢失，而是在整个用户会话中一直存在下去。

Session 的中文是“会话”的意思，在 Web 编程中 Session 代表了服务器与客户端之间的“会话”，意思是服务器和客户端在不断地交流。如果不使用 Session，则客户端每一次请求都是独立存在的，当服务器完成某次用户的请求后，服务器将不能再继续保持与该用户浏览器的连接。这样当用户在网站的多个页面间切换时(请求了多个页面)，页面之间无法传递用户的相关信息。这是因为，HTTP 协议是一种无状态(Stateless)的协议，利用 HTTP 协议无法跟踪用户。从网站的角度看，用户每一次新的请求都是单独存在的。

在 PHP 中，使用 $_SESSION[]可以存储特定用户的 Session 信息。并且，每个用户

的 Session 信息都是不同的。如果当前有若干个用户访问网站，则网站会为每个用户建立一个独立的 Session 对象，如图 9-11 所示。每个用户都无法访问其他用户的 Session 信息。因此一个用户访问网页时服务器为其创建的 Session 变量，别人是看不到的。

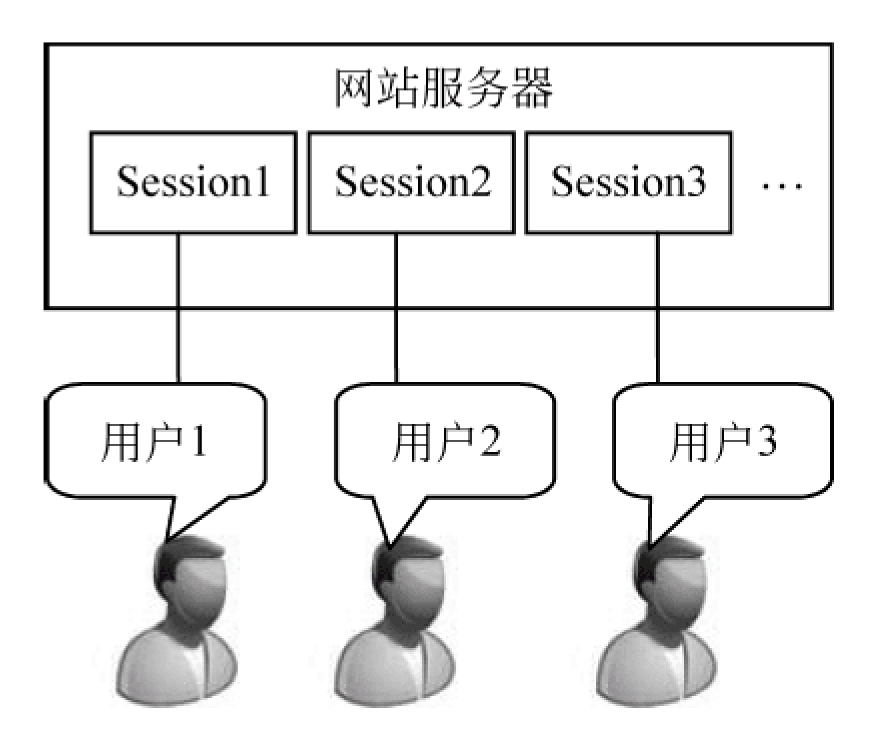


图 9-11　Session 示意图

9.3.1　存储和读取 Session 信息

在 PHP 中，使用 Session 前都需要在页面开头用 session_start()方法开启 Session 功能。然后就可利用 Session 变量存储信息了，它和使用普通变量存储信息很相似。语法如下：

```
$_SESSION["Session名称"]=变量或字符串信息
```

如果要读取 Session 变量信息，可将其赋给一个变量或直接输出，语法如下：

```
变量=$_SESSION["Session名称"]
```

为了验证 Session 变量能被网站中所有网页读取，可新建两个文件(或更多)，在 9-10.php中用 Session 变量存储信息，在 9-11.php 中读取 Session 变量信息。代码如下：

```
--------------------清单 9-10.php--------------------------
<?  session_start();                  //开启 Session
$_SESSION["username"]="小泥巴";        //将字符串信息存入 Session
$_SESSION["username"]="张三";          //修改 Session 变量值
$_SESSION["age"]=21;
$email='tang@163.com';
$_SESSION["email"]=$email;            //将变量信息存入 Session 变量
$_SESSION["user"]=array('name'=>'燕子','pwd'=>'111');        //将数组存入 Session 变量
?>
--------------------清单 9-11.php--------------------------
<?    session_start();                //开启 Session
echo $_SESSION["username"];           //输出"张三"
echo $_SESSION["age"];                //输出 21
echo $_SESSION["email"];      ?>
```

测试时首先运行 9-10.php 写入 Session 变量，再在同一个浏览器中输入 9-11.php 的网址，读取 Session 变量信息，则 9-11.php 的运行结果如下：

```
张三 21tang@163.com
```

这样就实现了通过 Session 变量在不同页面间传递数据。

说明：

(1) session_start()函数前面不能有任何代码输出到浏览器，最好加在页面头部，或先用 ob_start()函数打开输出缓冲区。

（2）对一个不存在的 Session 变量赋值，将自动创建该变量；给一个已经存在的 Session 变量赋值，将修改其中的值。

（3）如果新打开一个浏览器，去访问 9-11.php，则无法获取 Session 信息。因为新开一个浏览器相当于一个新的用户在访问。

（4）只要创建了 Session 变量，该 Session 变量就能被网站中所有页面访问，因此网站中任何页面（包括 9-10.php 自身）都能读取 9-10.php 创建的 Session 变量信息。

提示：在电子商务网站中常利用 Session 实现“购物车”，用户在一个页面中加入购物车的商品信息在转到另一个页面后仍然存在，这样用户可以在不同页面选择商品。所有商品的 id、价格等信息都保存在相应的 Session 变量中，直到用户去收银台交款或清空购物车时 Session 变量中的数据才被清除。由于服务器会为每个用户建立一个独立的 Session 对象，因此每个用户都有一辆专用的“购物车”。

9.3.2 Session 的创建过程和有效期

1. Session 的创建和使用过程

当用户请求网站中任意一个页面时，若用户尚未建立 Session 对象（如第一次访问），则服务器会自动为用户创建一个 Session 对象（它包含唯一的 Session ID 和其他 Session 变量），并保存在服务器内存中，不同用户的 Session 对象存储着各自特定的信息，

服务器将 Session ID 发送到客户端浏览器，而浏览器则将该 Session ID 保存在会话 Cookies 中。当浏览器再次向服务器发送 HTTP 请求时，会将 Session ID 信息一起发送给服务器。服务器根据该 Session ID 查找到对应的 Session 对象，就能识别出用户。这将有利于服务器对用户身份的鉴别，从而实现 Web 页面的个性化。

注意区分 Session 对象和 Session 变量，对于每个网站的访问者来说，网站都会为其建立一个 Session 对象，该 Session 对象中有一个 Session ID。如果程序中没有创建 Session 变量的代码，那么每个用户的 Session 对象中只含有 Session ID，否则，该 Session 对象中还包含许多个 Session 变量，如图 9-12 所示。也就是说，每个用户都有一个独立的 Session 对象，每个用户可以有 0 个到多个独立的 Session 变量。

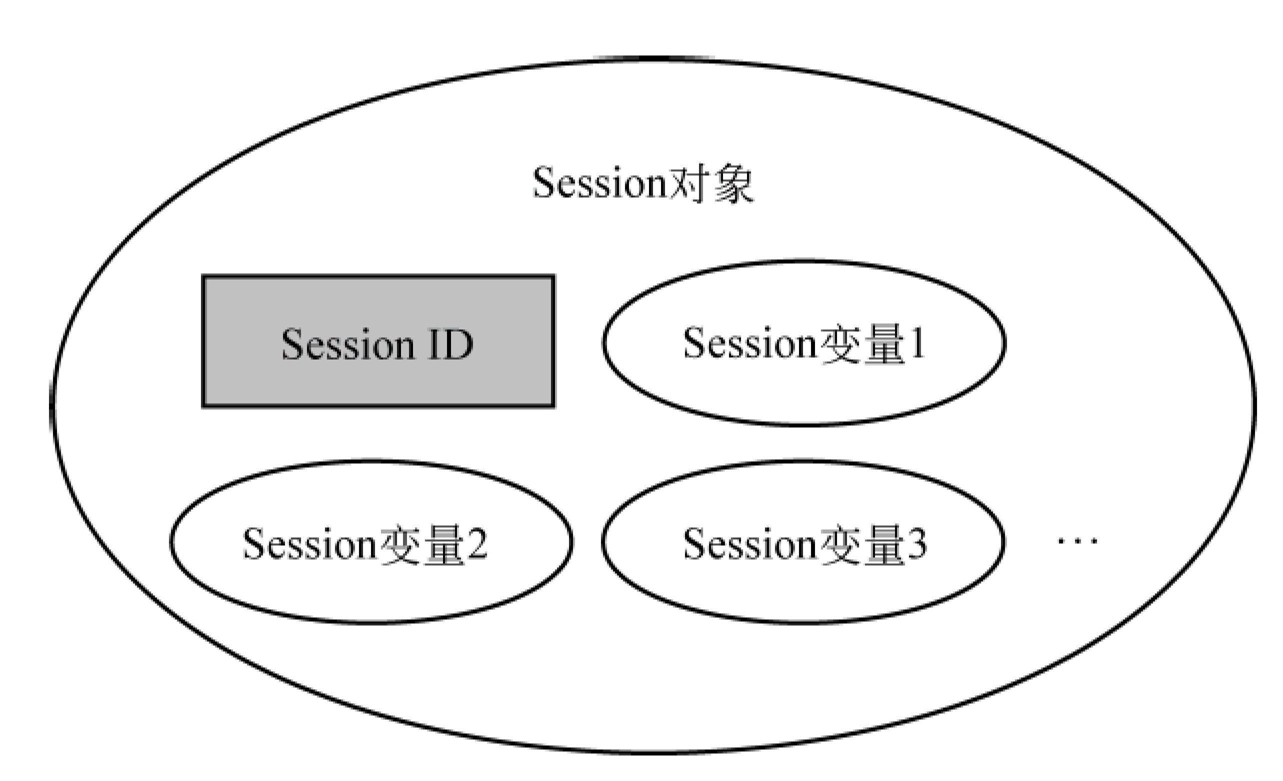


图 9-12　Session 对象和 Session 变量的关系示意图

下面的程序使用 session_id()函数返回用户的 Session ID 值。这验证了即使没有创建 Session 变量,用户仍然会拥有一个 Session ID。

```
<? session_start();
var_dump($_SESSION);          //输出 array(0) { },因为没有创建 Session 变量
echo session_id();            //输出 04c41641c2632c491c4d77d5898c0aa3
?>
```

提示:session_id()函数既可以获取当前 Session ID 值,也可以设置 Session ID 值,如 session_id('abc123'),此时必须在 session_start()函数调用之前使用。

最好不要把大量的信息存入到 Session 变量中,或者创建很多个 Session 变量。因为 Session 对象是要保存在服务器内存中的,而且要为每一个用户单独建立一个 Session 对象,如果保存的信息太多,同时访问网站的用户又很多,则如此多的 Session 对象是非常占用服务器资源的。

2. Session 的生命期

Session 对象的生命期是指从用户在 Session 有效期内第一次访问网站直到不再访问网站为止的这段时间。即一个 Session 开始于用户打开这个网站中的任意一个网页之时;结束于用户不再访问这个站点的那一刻,包括 Session 超时或主动删除 Session 两种情况。

注意:不再访问这个站点≠关闭浏览器。

关闭浏览器并不会使一个 Session 结束,因为服务器并不知道用户关闭了浏览器,但会使这个 Session 永远都无法访问到。因为当用户再打开一个新的浏览器窗口又会产生一个新的 Session。

3. 设置 Session 的有效期

Session 对象并不是一直有效的,它有个有效期,默认为 24 分钟(1 440 秒)。如果客户端超过 24 分钟没有刷新网页或访问网站中的其他网页,则该 Session 对象就会自动结束。不过可以修改 Session 对象的默认有效期,一种方法是在 PHP 的配置文件 php.ini 中修改系统默认值(session.gc_maxlifetime=1440),另一种方法是利用 ini_set()方法更改 Session 对象的默认有效期,代码如下:

```
session_start();
ini_set('session.save_path','/tmp/');              //设置保存路径
ini_set('session.gc_maxlifetime', 60);             //保存 1分钟
setcookie(session_name(), session_id(), time()+60, "/");
                                                   //设置会话 cookie的过期时间
```

提示:

(1) 虽然增加 Session 的有效期有时能方便用户访问,但这也会导致 Web 服务器内存中保存用户 Session 信息的时间增长,如果访问的用户很多,会加重服务器的负担。

(2) 不能单独对某个用户的 Session 设置有效期。

9.3.3 利用 Session 限制未登录用户访问

网站中有些页面要求只有登录成功的用户才能访问，如网站后台管理页面。利用 Session 可实现这种需求。具体方法是：在用户输入的用户名和密码验证通过后，用 Session 变量存储某些特征信息（如用户名），这个 Session 变量就相当于“票”，然后在其他对安全性有要求的页面最前面检查这些 Session 变量是否存在（即验票），如果这些特征值为空，表示没有经过合法认证，是通过直接输入网页的网址进入的，就拒绝其访问。示例代码如下。

```
----------------------清单 9-12.php----------------------------
<? session_start();
if(isset($_POST["submit"]))  {      //判断是否单击了登录按钮
    $user=$_POST["userName"];
      $pw=$_POST["PW"];
if($user=="admin" &&$pw=='123'){ //判断用户名密码是否正确
    $_SESSION['user']=$user;      //将用户名存入$_SESSION['user'],这是关键
    header('Location:9-15.php');}
else    echo "用户名或密码错误";
}
else  echo '
<form method="post" action="">
  用户名:<input type="text" name="userName" />
  密 码:<input type="password" name="PW" />
  <input name="submit" type="submit" value="登录" />
</form>';      ?>
----------------------清单 9-13.php----------------------------
<? session_start();
if(isset($_SESSION['user']))        //如果$_SESSION['user']不为空
    echo "欢迎您,".$_SESSION["user"]."<br/>
        <a href='9-16.php?action=logout'>注销</a>";
else
    echo "未登录用户不允许访问";      ?>
```

程序运行效果是：如果用户没有经过 9-12.php 页面登录或登录失败，而是直接运行 9-13.php，就会提示“未登录用户不允许访问”；如果在 9-12.php 登录成功过，则以后每次运行 9-13.php 都会显示欢迎信息。

说明：

(1) 该实例必须先运行 9-12.php，以对登录成功用户赋予 $_SESSION['user'] 变量，而 9-13.php 用来检查该 Session 变量是否为空，请注意 9-13.php 中并没有采用 $_POST 获取表单变量，而是读取和输出 9-12.php 中创建的 Session 变量。

(2) 9-12.php 中创建的 Session 变量可以被网站中的所有网页访问。因此可以将

9-13.php 中的代码放到网站中所有对安全性有要求的网页的最前面。

9.3.4 删除和销毁 Session

删除 Session 常用来实现用户注销的功能，使得用户能够安全退出网站。在 PHP 中，使用 unset()方法可以删除单个 session 变量。使用 session_unset()函数可删除当前内存中 $_SESSION 数组中的所有元素，它等价于 $_SESSION＝array()或 unset($_SESSION)。例如：

```
<? session_start();
unset($_SESSION["username"]);      //删除$_SESSION 中一个 session 变量
session_unset();                   //删除$_SESSION 中所有 session 变量
?>
```

但是，session_unset 只能删除 $_SESSION 数组中的所有元素，并不能删除对应的 Session ID，也不能删除保存 Session ID 的文件。而 session_destroy()函数就能删除 Session ID，并销毁 Session 文件，但它不会删除内存中的 $_SESSION 数组中的所有元素。例如：

```
<? session_start();
    echo '<p>这个用户的 Session 编号为'.session_id().'</p>';
    $_SESSION["user_name"]="布什";
    session_destroy();          //清除 Session ID
    echo '<p>这个用户的 Session 编号为'.session_id().'</p>';
    echo $_SESSION["user_name"]; //会输出"布什"
?>
```

运行结果为：

```
这个用户的 Session 编号为 43abc321fc0e76486f32dd86fabde568
这个用户的 Session 编号为
布什
```

因此，如果要彻底删除 Session，实现用户安全注销功能，可以将 session_unset()与 session_destroy()函数结合使用，并且还需清除浏览器中的会话 Cookie 信息，这可以通过调用 setcookie 函数将会话 Cookie 设置为过期即可。下面是一个注销用户登录的例子。

```
---------------------清单 9-14.php----------------------
<?
if($_GET['action']=="logout"){
session_start();                    //启动会话
setcookie("user","",time()-60);     //将会话 Cookie 变量 user 设置为过期,即删除 Cookie
session_unset();                    //删除$_SESSION 中的 Session 变量
session_destroy();                  //销毁 Session,删除 Session ID
```

```
header("Location:9-14.php");        //回到登录界面
}    ?>
```

9.4 使用 $ _COOKIE 读取 Cookie

使用 Session 只能让网站记住当前正在访问的用户,但有时网站还需要记住曾经访问过的用户,以便在用户下次访问时,提供个性化的服务。这就需要用到 Cookie 技术。Cookie 能为网站和用户带来很多好处,比如它可以记录特定用户访问网站的次数、最后一次访问时间、用户在网站内的浏览路径,以及使登录成功的用户下次自动登录等。

也有一些 Cookie 的高级应用,如在购物网站浏览商品页面时,该网站程序可以将用户的浏览历史记录到 Cookie 中,当用户下次再访问时,网站根据用户过去的浏览情况为用户推荐感兴趣的内容。

Cookie 实际上是一个很小的文本文件,网站通过向用户硬盘中写入一个 Cookie 文件来标识用户。当用户下次再访问该网站时,浏览器会将 Cookie 信息发送给网站服务器,服务器通过读取以前写入的 Cookie 文件中的信息,就能识别该用户。

Cookie 有两种形式:会话 Cookie 和永久 Cookie。前者是临时性的,只在浏览器打开时存在(存储在用户机器的内存中),主要用来实现 Session 技术;后者则永久地存放在用户的硬盘上并在有效期内一直可用。Cookie 文件默认保存在“C:\Documents and Settings\登录用户名\Cookies”文件夹中。

在 PHP 中,利用 setcookie()函数可以创建和修改 Cookie,以及设置 Cookie 的有效期;而使用 $ _COOKIE[]数组可以读取 Cookie 变量的值。

9.4.1 创建和修改 Cookie

创建 Cookie 最简单的方法是使用 setcookie()函数。语法如下:

```
setcookie(name, value, expire, path, domain, secure)
```

其中,name 用来定义一个 cookie 的变量名,value 用来设置 Cookie 变量值,expire 用来定义 Cookie 的有效期;而 path、domain、secure 分别用来规定 Cookie 的有效目录、有效域名和是否采用 HTTPS 来传输 Cookie,这三个参数不常用。除了 name 和 value 是必需的参数外,其他参数都是可选的。

下面是一个创建 Cookie 的程序:

```
----------------------清单 9-15.php--------------------------
<?
setcookie('tmpcookie','这是个临时 cookie');      //不设置过期时间
setcookie('userName','小泥巴',time()+60);       //设置过期时间为 60 秒,永久 cookie
setcookie('age',21,time()+60);
setcookie('sex','女',time()+60,'','',false);     //设置 setcookie 的所有参数
?>
```

上例中设置了 4 个 Cookie 变量，变量名分别为 tmpcookie、userName、age 和 sex。其中 tmpcookie 没有设置过期时间，因此它仅仅是个会话 Cookie，会话 Cookie 并没有保存到文本文件中，关闭浏览器后，tmpcookie 将立即失效。而其他 3 个 Cookie 均设置了过期时间，因此是永久 Cookie，它们将在关闭浏览器 1 分钟后失效。

要查看 9-15.php 写入的 Cookie 文件，可以打开保存在"C:\Documents and Settings\登录用户名\Cookies"目录下的 Cookie 文件，文件内容如图 9-13 所示。



图 9-13　客户端 Cookie 文件中的信息

由此可见，永久 Cookie 变量均保存在了客户端的 Cookie 文件中，而会话 Cookie 没有保存。Cookie 变量名和变量值中如果含有中文或特殊字符，会自动经 urlencode()函数处理转换成 gb2312 编码形式。

如果要修改 Cookie 变量的值，可以用 setcookie()函数给变量重新赋值。例如：

```
setcookie('age',24,time()+60);    //将 Cookie 变量 age 的值修改为 24
```

但修改 Cookie 时设置过期时间的参数不能省略，否则该 Cookie 会被修改成临时 Cookie。

提示：

(1) 在使用 setcookie()函数前，不要有任何 HTML 内容输出到浏览器，因为 Cookie 也是作为 HTTP 协议头的一部分，否则 setcookie()创建 Cookie 将失败。

(2) Cookie 变量的值总是字符串数据类型。

(3) 在 PHP 中，还能使用 header 函数设置 Cookie。例如：

```
header("Set-Cookie: nickname=小泥巴; expires=".gmstrftime("%A, %d-%b-%Y %H:
%M:%S GMT", time()+(86400 * 30)));
```

其中 nickname 是 Cookie 变量名，"小泥巴"是 Cookie 变量值，expires 用来设置过期时间。Set-Cookie 参数和 expires 参数之间用"；"号隔开。

9.4.2　读取 Cookie

在客户端写入 Cookie 后，当用户再次向网站发送 HTTP 请求时，就会将 Cookie 信息放在 HTTP 请求头中一起发送给服务器，服务器会自动获取 HTTP 请求头中的 Cookie 信息，并将这些信息保存到 $_COOKIE 数组中。因此通过 $_COOKIE 可以读取所有从客户端传过来的 Cookie 信息。下面的程序用来读取 9-15.php 中创建的 Cookie 信息。

---------------------清单 9-16.php-------------------------

```
<?
```

```
$user=$_COOKIE['userName'];
$age=$_COOKIE['age'];
$sex=$_COOKIE['sex'];
echo $user.$age.'岁,性别'.$sex;    ?>
```

为了测试该程序,首先运行 9-15.php 创建 Cookie,再运行 9-16.php 读取 Cookie,则 9-16.php 的运行结果如下:

```
小泥巴 21岁,性别女
```

说明:由于 Cookie 存放在了硬盘中,因此即使重启电脑后,再打开浏览器访问 9-18.php 也能读取到 Cookie(只要 Cookie 没过期)。

9.4.3 Cookie 数组

实际上,使用 setcookie()函数还可以创建 Cookie 数组。例如:

```
<?
setcookie("user[name]","张三",time()+600);
setcookie("user[id]","zhang3",time()+600);
setcookie("user[sex]","男",time()+600);
setcookie("user[age]",23);    ?>
```

创建 Cookie 数组时,对于数组元素的索引可以是整数或字符串,但索引两边不要用引号(如 user[id]不能写成 user['id']),因为 PHP 会自动给 setcookie 中的数组索引加引号。

要读取 Cookie 数组,可使用循环语句遍历数组,也可单独输出数组元素。代码如下:

```
<?
foreach($_COOKIE['user'] as$key=>$value){
    echo $key.'=>'.$value.'  ';}        //输出 Cookie 数组中所有元素
    var_dump($_COOKIE);                 //输出$_COOKIE 中的所有内容
    echo  $_COOKIE['user']['name'];     //输出 Cookie 数组中一个元素
?>
```

运行结果如下(必须先运行创建 Cookie 数组的程序):

```
name=>张三 id=>zhang3 sex=>男 age=>23 张三
```

9.4.4 删除 Cookie

有时用户可能希望网站不再记住自己过去访问的信息,这时可以删除 Cookie。删除 Cookie 有两种方法:一是将 Cookie 的变量值设置为空,并且不设置有效期(不设置有效期将删除 Cookie 文件中的 Cookie 变量);二是将 Cookie 的有效期设置为过去的某个时间。无论使用哪种方法,浏览器接收到这样的 Cookie 响应头信息后,将自动删除用户硬盘中的 Cookie 文件和内存中的 Cookie 信息。例如,下面程序的功能是将 9-15.php 中创建的 Cookie 全部删除。

```
<?
setcookie('userName','');              //删除 Cookie的方法 1
setcookie('age',21,time()-600);        //删除 Cookie的方法 2
setcookie('sex','女',time()-600);
var_dump($_COOKIE);                    //用来查看上述 Cookie数组元素是否已经删除
?>
```

9.4.5　Cookie 程序设计举例

如果要编写 Cookie 应用的程序，一般的流程是：首先尝试获取某个 Cookie 变量，如果有，则表明是老客户，读取其 Cookie 信息，为其提供个性化的服务；如果没有，则表明是第一次来访的新客户，通过表单获取其身份信息，再将这些信息存入到 Cookie 变量中去。

1. 用户自动登录

用户自动登录程序的实现流程如图 9-14 所示。

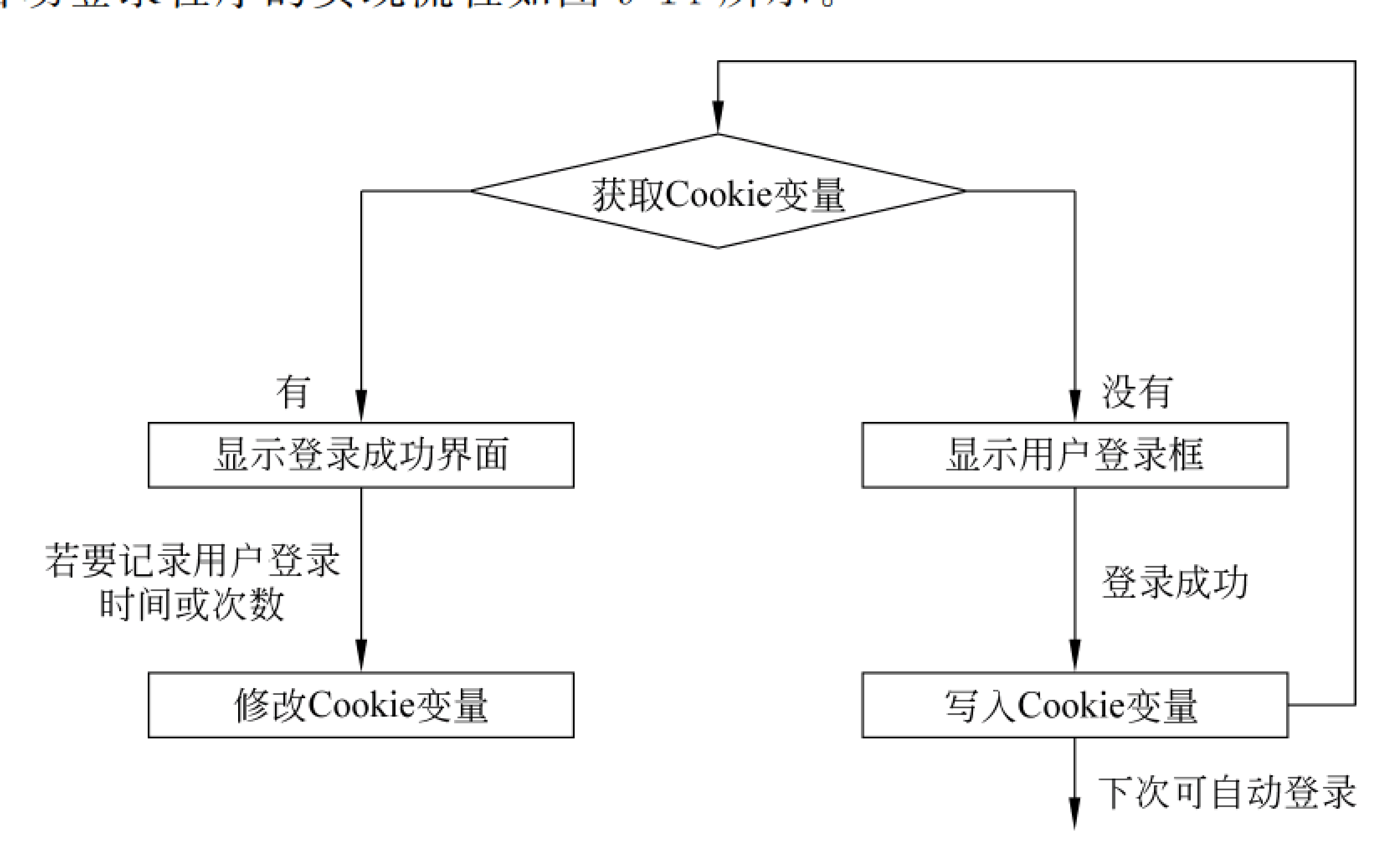


图 9-14　用户自动登录程序的一般流程图

在下面的实例中，如果用户第一次访问 9-17. php，则会显示如图 9-15 所示的登录表单，如果用户登录成功并选择了保存 Cookie，则以后再次访问 9-17. php 时就不需要登录，会自动转到欢迎界面（见图 9-16），并显示用户的访问次数和上次登录时间。该实例包括两个文件，其中 9-17. php 是主程序，9-18. php 用于获取表单信息并写入 Cookie 等，代码如下：

```
----------------------清单 9-17.php-----------------------
<?
if($_COOKIE["user"]["xm"]<>""){                          //尝试获取指定的 Cookie变量，如果有
    $visnum=intval($_COOKIE["user"]["num"])+1;     //将原来的访问次数加 1
    $expire=intval($_COOKIE["user"]["expire"]);    //获取有效期
    //将本次访问时间写入 Cookie
    setcookie("user[dt]",date("Y-m-d h:i:s"),time()+3600*$expire);
```

```
        setcookie("user[num]",$visnum,time()+3600*$expire);
                                              //将本次访问次数写入 Cookie
        echo"欢迎您:".$_COOKIE["user"]["xm"];   //输出 Cookie变量的值
        echo "<br/>这是您第".$visnum."次访问本网站";
        echo "<br/>您上次访问是在".$_COOKIE["user"]["dt"];
}
else              //没有 Cookie则显示登录表单
        echo '<html><body>
<div style="border:1px solid #06f; background:#bbdeff">
  <form method="post" action="9-20.php" style="margin:4px;">
    <p>账号:<input name="xm" type="text" size="12"></p>
    <p>密码:<input name="Pwd" type="password" size="12"></p>
    <p>保存:<select name="Save">
        <option value="-1">不保存</option>
        <option value="7">保存 1周</option>
        <option value="30">保存 1月</option></select>
        <input type="submit" value="登 录"></p>
  </form></div></body></html>'    ?>
---------------------清单 9-18.php--------------------------
<? if($_POST["xm"]=="admin" &&$_POST["Pwd"]=="123")  {
setcookie("user[xm]",$_POST["xm"],time()+3600*intval($_POST['Save']));
setcookie("user[dt]",date("Y-m-d h:i:s"),time()+3600*$expire);
    //写入 Cookie
setcookie("user[num]",1,time()+3600*intval($_POST['Save']));
    //保存有效期到 Cookie
setcookie("user[expire]",$_POST['Save'],time()+3600*intval
($_POST['Save']));
echo $_POST["xm"].": 首次光临";
    //var_dump($_COOKIE);  }
else
        echo "<script>alert('用户名或密码不对');location.href='9-17.php';
</script>";
?>
```



图 9-15 9-17. php 的第一次运行结果



图 9-16 9-17. php 登录成功后

2. 记录用户的浏览路径

在电子商务网站中，经常需要记录用户的浏览路径，以判断用户对哪些商品特别感兴趣或哪些商品之间存在销售关联。下面的例子使用 Cookie 记录用户浏览过的历史页面。该网站将每个页面的标题保存在该页面的 $title 变量中，用户每访问一个页面就会将新访问页面的标题添加到 Cookie 变量 $_COOKIE["history"]值中。随着用户访问页面的增多，该 Cookie 变量中保存的含有页面标题的字符串会越来越长。将该 Cookie 变量切分成数组，然后输出数组元素的值就输出了用户最近访问页面的标题列表。

```
----------------------清单 9-19.php(商品页)----------------
<?      ob_start();      //打开缓冲区,以便在有输出后还能设置 Cookie
$title="西游记"          //商品页有很多,其他商品页的 title 是水浒传、西游记等 ?>
<html><head>
        <title><?=$title ?></title>      </head>
<body>      <h3 align="center"><?=$title ?>商品页面</h3>
<p>同类商品:<a href="hlm.php">红楼梦</a><a href="shz.php">水浒传</a>
<a href="sg.php">三国演义</a></p>
<? require("9-20.php") ?>
</body></html>
----------------------清单 9-20.php(商品页调用的记录浏览历史的程序)------
<?$history=$_COOKIE["history"];              //获取记录浏览历史的 Cookies
    if($history=="")                         //如果浏览历史为空
        $path=$title;                        //将当前页的标题保存到 path 变量中
    else
        $path=$title."/".$history;           //将当前页的标题加到浏览历史的最前面
    //将$path 保存到 Cookie 变量中,设置过期时间为 30 天
    setcookie("history",$path,time()+30*3600);
    $arrPath=explode("/",$path);             //将$path 分割成一个数组$arrPath
echo "您最近的浏览历史:<hr/>";
foreach ($arrPath as$key=>$value){
        if($key>9) break;                    //只输出最近的 10 条
        echo ($key+1) .". ".$value ."<br/>"; //输出浏览历史
}?>
```

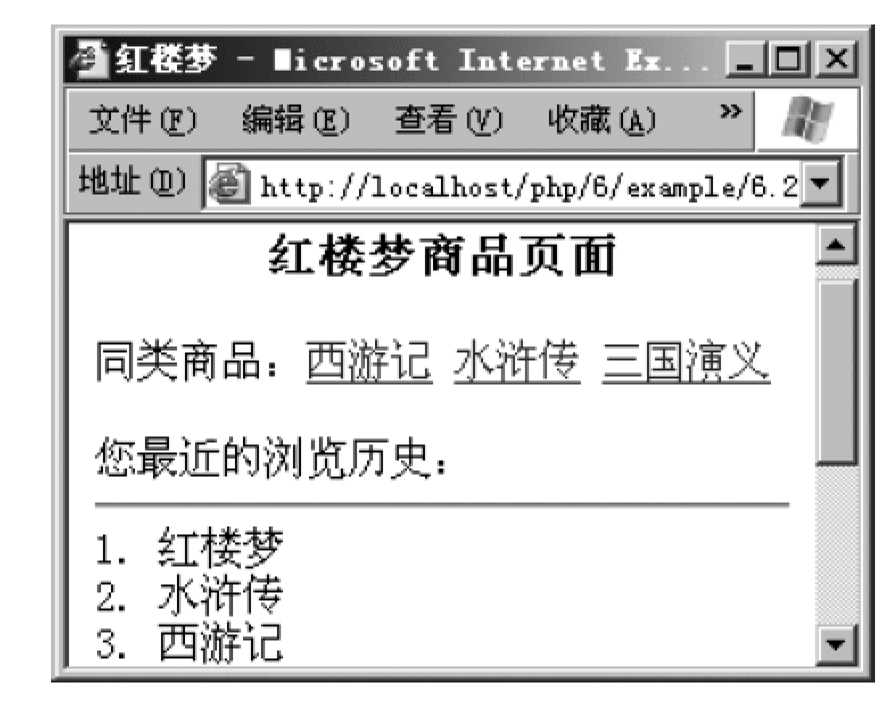


图 9-17　9-19.php 的运行结果

说明：测试时应首先将 9-19.php 重命名成几个文件(xyj.php、shz.php、hlm.php、sg.php)，然后将这几个文件第 2 行中的 $title 变量值分别改成“水浒传”“西游记”和“三国演义”。接下来运行其中任何一个文件，再通过单击链接转到其他文件，会发现每浏览一个页面，它的标题就会记录到浏览历史中，如图 9-17 所示，而且关闭浏览器后再打开，浏览历史依然不会丢失，从而基本实现了保存用户

浏览历史的目的。

当然，在实际的电子商务网站中，记录浏览历史还会将用户的浏览历史保存到服务器的数据库中，那样浏览历史能更长久地保存，网站还能根据所有用户的浏览历史进行数据分析和统计。

9.5 使用 $ _FILES 获取上传文件信息

文件上传是网站程序的一项基本功能，例如有些网站允许用户上传图片文件、上传文档(Word 或 PPT)等。PHP 可轻松实现将本地文件上传到 Web 服务器的功能。

在 PHP 中，文件上传功能的实现步骤为：

(1) 在网页中添加文件上传表单，单击“上传”按钮后，选择的文件数据将发送到服务器；

(2) 用 $ _FILES 获取与上传文件有关的各种信息；

(3) 用文件上传处理函数对上传文件进行后续处理。

9.5.1 添加上传文件的表单

在 HTML 中，可以使用表单中的文件上传域来提交要上传的文件。下面是一个上传文件的 HTML 表单代码，它在浏览器中的显示效果如图 9-18 所示。

```
------------------清单 9-21.php---------
<h3 align="center">上传文件的演示实例</h3>
<p>请选择要上传的 jpg 图片文件</p>
<form action="9-22.php" method="post" enctype="multipart/form-data">
   <input type="file" name="upfile" /><br><br>
   <input type="submit" value=" 上传 " />
</form>
```

说明：如果表单中有文件上传域，则定义表单时必须设置 enctype = " multipart/form-data ", method 属性必须设置为 post(这是因为 get 方式发送的数据量不能超过 8KB)。action="9-22. php"表示上传的数据将发送给 9-22. php，因此 9-22. php 是处理上传文件的脚本。

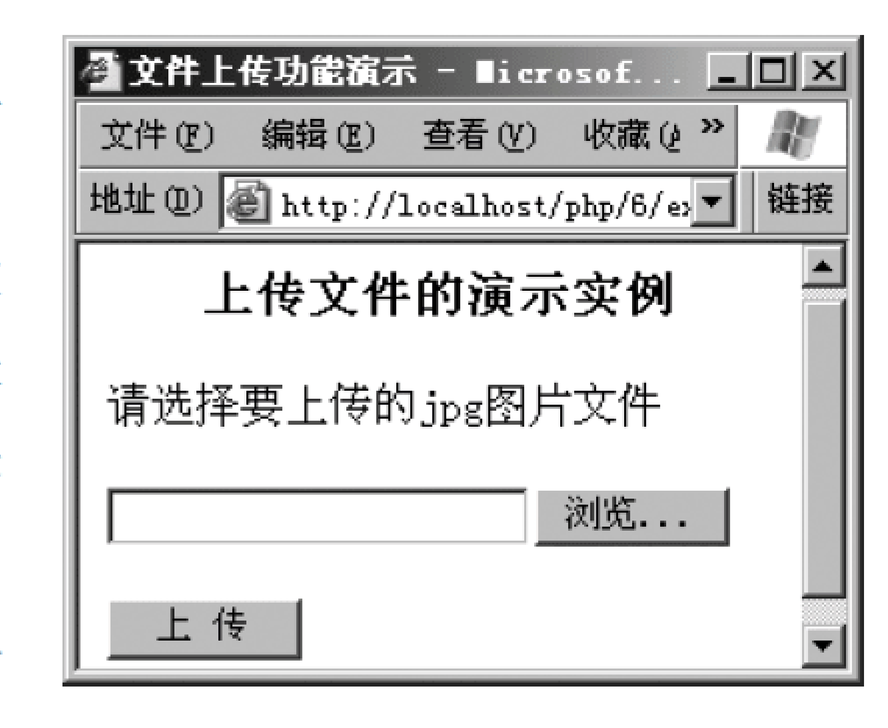


图 9-18 上传文件的网页 9-21. php

提示：如果要限制上传文件的大小，可以在表单中添加一个隐藏域：<input type = "hidden" name="MAX_FILE_SIZE" value="10240">，并且该隐藏域必须放在文件上传域的前面，否则会设置失效。

9.5.2　使用 $_FILES 获取上传文件信息

当用户单击“上传”按钮后，上传的文件数据将发送给服务器。在处理脚本 9-22.php 时，可以使用 $_FILES 来获取上传文件的信息。$_FILES 是一个多维数组，它可以保存所有上传文件的信息，以及上传过程中的错误信息。如果文件上传域的 name 属性值为 upfile，则可以使用 $_FILES['upfile']来访问上传文件的有关信息。$_FILES['upfile']是一个一维数组，数组元素是上传文件的各种属性，具体如下：

$_FILES['upfile']['name']——客户端上传文件的原名称(不包括路径)。

$_FILES['upfile'] ['type']——上传文件的 MIME 类型，如 image/gif 等。

$_FILES['upfile'] ['size']——已上传文件的大小，单位是字节。

$_FILES['upfile'] ['tmp_name']——上传文件在服务器端保存的临时文件名(包含路径名)。

$_FILES['upfile'] ['error']——上传文件出现的错误号，是一个整数。

我们可以在 9-22.php 中输入如下代码，来输出 $_FILES['upfile']数组的内容。

```
<? var_dump($_FILES['upfile']);   ?>
```

选择上传文件 guangxue.gif，单击上传按钮后，执行结果如下。

```
array(5){["name"]=>string(12) "guangxue.gif" ["type"]=>string(9) "image/gif" ["tmp_name"]=>string
(26) "C:\WINDOWS\TEMP\php11C.tmp" ["error"]=>int(0) ["size"]=>int(44863) }
```

9.5.3　保存上传文件到指定目录

通过上述步骤，上传文件已经保存在了 C:\Windows\temp\目录下，文件名为 php11C.tmp。接下来，必须将上传文件移动到网站目录下的指定目录中，并为它重命名，以便让网站目录内的网页可以引用该文件。

要移动文件到指定目录，一般使用 move_uploaded_file()函数。该函数的语法如下：

```
move_uploaded_file(文件原来的路径和文件名,文件的目的路径和文件名)
```

提示：该函数还提供了一个额外的功能，即检查并确保由第一参数指定的文件是合法的上传文件(即通过 HTTP POST 上传机制所上传的)，这对于网站安全是至关重要的。即该文件包含了 is_uploaded_file()函数的功能。

下面是 9-22.php 的代码。该程序的功能是将上传的临时文件移动到网站指定目录内，并为它重命名。为此先要检查指定目录是否存在，如果不存在，则创建，再用当前时间生成文件名。最后通过 $_FILES 获取临时文件的文件名作为原文件名，就可以用 move_uploaded_file()将临时文件移动到指定目录下了。运行结果如图 9-19 所示。

```
-----------------清单 9-22.php--------------------
<?          //$upload_dir是上传文件的目录,getcwd()可获取当前脚本所在目录
$upload_dir=getcwd() . "\\images\\";      //即“当前目录\images”
if(!is_dir($upload_dir))                  //如果目录不存在,则创建
```

```
    mkdir($upload_dir);
function makefilename() {                  //此函数用于根据当前时间生成上传文件名
    $curtime=getdate();                    //获取当前系统时间,生成文件名
    $filename=$curtime['year'] .$curtime['mon'] .$curtime['mday'] .$curtime['hours'] .$curtime['
minutes'] .$curtime['seconds'] . ".jpg";
        return $filename;                  //返回生成的文件名
    }
$newfilename=makefilename();
$newfile=$upload_dir .$newfilename;        //生成文件路径名加文件名
if(file_exists($_FILES['upfile']['tmp_name'])) {
                                           //如果这个临时文件存在,表明上传成功
    move_uploaded_file($_FILES['upfile']['tmp_name'],$newfile);
    echo "客户端文件名: " .$_FILES['upfile']['name'] . "<br>";
    echo "文件类型: ".$_FILES['upfile']['type']. " ";
    echo "大小: " .$_FILES['upfile']['size'] . "字节<br>";
    echo "服务器端临时文件名: " .$_FILES['upfile']['tmp_name'] . "<br>";
    echo "上传后的文件名: " .$newfile . "<br>";
    echo '文件上传成功 [<a href="#" onclick="history.go(-1)">继续上传</a>]
            <p>右边是上传的图片文件:
        <img src="images/'.$newfilename .'"></p>';}             //用 img 标记显示上传的图片
    else        echo "上传失败,错误类型:".$_FILES['upfile']['error'];
?>
```



图 9-19 9-22.php 的运行结果

9.5.4 同时上传多个文件

多个文件上传和单文件上传实现的方法是相似的,只需要在表单中提供多个文件上传域,并指定 name 属性值为同一个数组即可。例如,在下面的程序中,用户可以选择三个本地文件一起上传给服务器,客户端页面代码(9-23.php)如下,显示效果如图 9-20 所示。

--------------------清单 9-23.php--------------------

```
<h3 align="center">多文件上传功能演示</h3>
<p>请选择要上传的三张图片文件</p>
<form action="9-24.php" method="post" enctype="multipart/form-data">
    文件 1:<input type="file" name="upfile[]" /><br><br>
    文件 2:<input type="file" name="upfile[]" /><br><br>
    文件 3:<input type="file" name="upfile[]" /><br><br>
  <input type="submit" value=" 上 传 " />
</form>
```

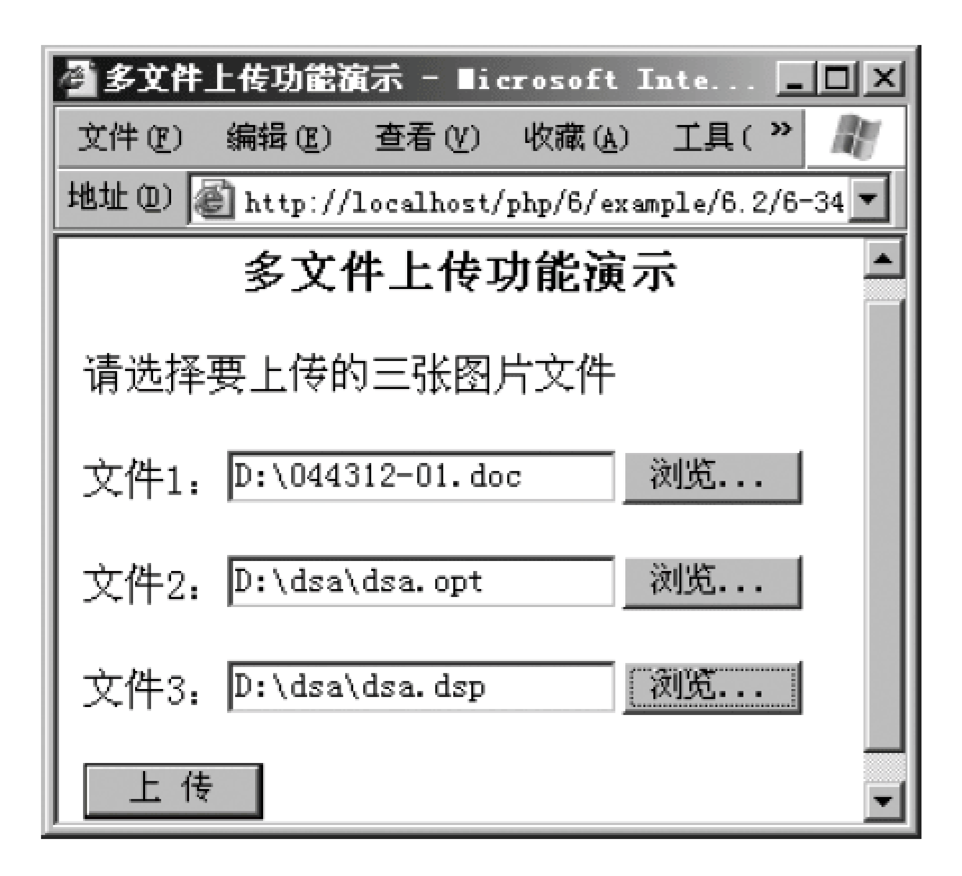


图 9-20　多文件上传程序界面

在上面的代码中,将三个文件上传域以数组的形式组织在一起,当表单提交给脚本文件 9-24.php 时,服务器端同样可以使用 $_FILES 获取所有上传文件的信息,但 $_FILES 已经由二维数组转变成了三维数组。例如,保存第一个文件的文件名的数组元素是 $_FILES['upfile']['name'][0]。

接下来,根据 $_FILES 获取的临时文件文件名,同样可以用 move_uploaded_file() 将临时文件移动到指定目录下,就实现了多文件的上传。

习　题　9

一、练习题

1. 下列有关 get 和 post 方法传递信息的说法中,正确的是(　　)。
   A. get 方法是通过 URL 参数发送 HTTP 请求,传递参数简单,且没有长度限制
   B. post 方法是通过表单传递信息,可以提交大量的信息
   C. 使用 post 方法传递信息会出现页面参数泄露在地址栏中的情况
   D. 使用 URL 可以传递多个参数,参数之间需要用"?"连接
2. 下列哪个数组不可能用来获取表单元素的值?(　　)
   A. $_REQUEST[]　　B. $_POST[]
   C. $_GET[]　　D. $_SERVER[]
3. 下列哪个函数不是缓冲区操作函数?(　　)

A. ob_flush()　　　　B. flush()

C. ob_flush_clean()　　　　D. ob_end_clean()

4. 下面程序段执行完毕，页面上显示的内容是什么？（　　）

```
<?=htmlspecialchars("<a href='http://www.sohu.cn'>搜狐</a>") ?>
```

A. 搜狐

B. 搜狐

C. 搜狐（超链接）

D. 该句有错，无法正常输出

5. 关于 Session 和 Cookie 的区别，下列哪项是错误的？（　　）

A. 服务器会自动为用户建立 Cookie 对象

B. 用户关闭浏览器，网站为该用户创建的 Session 对象将无法访问

C. 用户新开一个浏览器窗口，网站为其创建一个新的 Session 对象

D. 用户关闭计算机，其 Cookie 仍然存在

6. 如果要删除 Cookie，可以使用下列哪个函数？（　　）

A. clearcookie()　　　　B. setcookie()

C. destroy()　　　　D. ob_end_flush()

7. 在 PHP 中要使用 Session，必须先调用下列哪个函数？（　　）

A. ob_start()　　　　B. session_id()

C. session_start()　　　　D. setcookie

8. 有些语句要求只有在服务器还没有向浏览器输出任何信息前才能使用，下列语句中无此要求的是（　　）。

A. setcookie('userName','');　　　　B. session_start();

C. header("location:9-8. php ");　　　　D. session_unset();

9. 在网站页面之间传递值的方法有（　　）（多选）。

A. Session 变量　　B. Cookie 变量　　C. 表单变量　　D. URL 变量

如果超链接的地址是 http://ec. hynu. cn/instr. php? abc=3&bcd=test，要获取参数 bcd 的参数值应使用的命令是________。

10. Session 对象默认情况下有效期是________分钟。要提前结束一个 Session，可以用________方法。要返回 Session 对象的 id，可以用________函数。

11. 在 A 网页上创建了一个 Session 变量：$_SESSION["user"]="张三"，在 B 网页上要输出这个 Session 变量的值，应使用________。

12. 假设用 $_POST['username']能获取到信息，则能判断提交给该页的表单中含有________属性为 username 的表单元素。该表单 form 标记的 method 属性为________。

13. 如果要修改一个 Cookie 变量的有效期，需使用________函数。

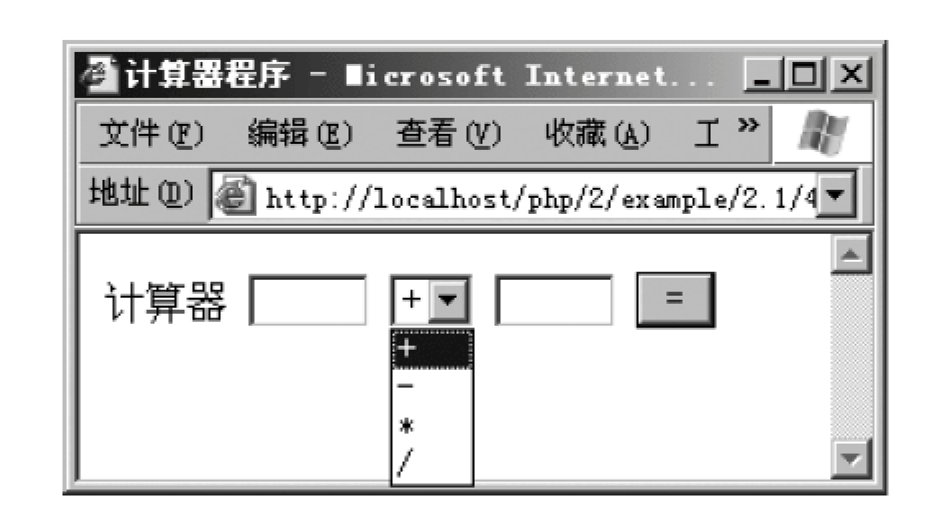
14. 要获取上传文件的信息，需要使用________数组，要将上传文件移动到指定位置，需要使用________函数。

15. 在 PHP 中有哪些常用的超全局变量？请简述它们的主要功能。

16. 在 form 标记中，method 和 action 属性的作用分别是什么？
17. 若要在 PHP 中快速获取一组复选框的值，应如何命名这些复选框？
18. 在 PHP 中，设置 URL 参数的方法有哪些？
19. 能否将多个不同的表单页提交给同一个表单处理页？

二、编程题

1. 编写一个简单计算器程序，在表单中添加两个文本框供用户输入数字，下拉框用来选择运算符，当单击“＝”按钮后在网页上输出结果，如图 9-21 所示。要求：单击“＝”按钮后用户在文本框中输入的数字仍然存在。



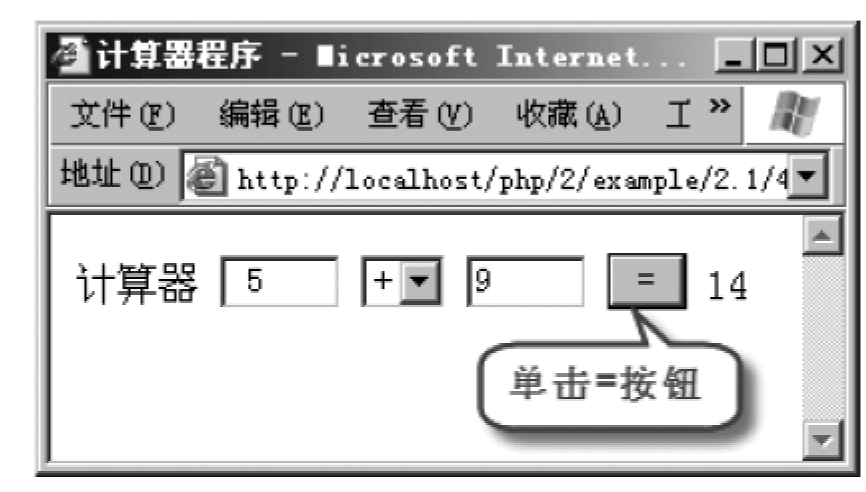


图 9-21　计算器程序效果图

2. 下面的表单会向服务器发送哪几个变量信息？编写服务器端程序获取发送来的所有变量信息并按以下格式输出：××您好，您住在××，您的密码是×××。

```
<form name="abc" method="post" action="g4.php">
  用户名：张三<input type="radio" name="user"  value="张三">
  李四：<input type="radio" name="user"  value="李四">
  住址：    <select name="addr">
        <option value="长沙">长沙</option>
      <option value="衡阳">衡阳</option>
</select>
  <input type="hidden" name="pwd" id="hi" value="123">
      <input type="submit" value="登录">
</form>
```

3. 下面是一个获取表单提交信息的程序，名称为 rec.php，请写出一个能让它获取数据的表单代码。

```
<?  $name=$_POST["name"];    //获取各个表单元素的值
    $Sex=$_POST["Sex"];
    $hob=$_POST["hobby"];
    $car=$_POST["career"];
?>
```

4. 编写 PHP 程序产生一个随机数，并让用户在文本框输入数字来猜测该随机数(见图 9-22)，用户有 5 次机会，根据用户的猜测结果给予相应提示(提示：将程序在猜测前产生的随机数保存在表单隐藏域中，这样用户每次猜测时该随机数都不会发生变化)。该程

序的表单代码如下，请补充 PHP 代码。



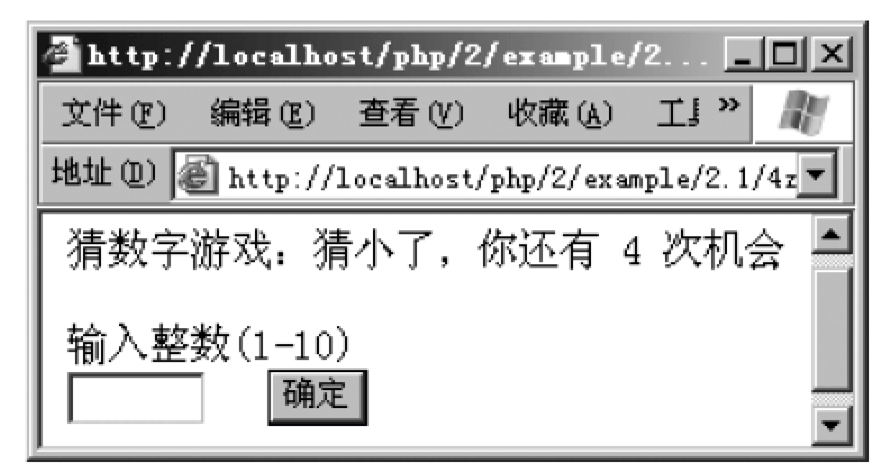


图 9-22 猜数字游戏程序效果图

```
<form method="post" action="">      输入整数(1-10)<br />
    <input type="text" name="SZ" size="6">
    <input name="rand" type="hidden" value="<?=$a ?>" />
                                        <!--保存猜测前产生的随机数-->
    <input name="last" type="hidden" value="<?=$b ?>" />
                                        <!--保存剩余机会次数-->
    <input type="submit" name="sub" value="确定">
</form>
```

5. 编写回答多项选择题的 PHP 程序。程序界面如图 9-23 所示。如果输入正确答案(PHP、ASP、JSP)，则在网页上提示“正确”，如果少选了，则提示“回答不全”，否则提示“错误”。



图 9-23 回答多项选择题的程序界面

第10章 PHP访问数据库



10.1　MySQL 数据库的使用

MySQL 是一种流行的关系型数据库管理系统软件。与其他数据库管理系统(如 Oracle、DB2、SQL Server)相比,MySQL 具有体积小、速度快、功能齐全并且完全免费等特点,使得一般中小型 PHP 网站的开发都选择使用 MySQL 作为网站数据库。

10.1.1　数据库基础

所谓数据库,就是按照一定数据模型组织、存储在一起的,能为多个用户共享的,与应用程序相对独立、相互关联的数据集合。

目前绝大多数数据库采用的数据模型都是关系数据模型,所谓"关系",简单地说就是二维表。所以,数据库在逻辑上可以看成是一些表格组成的集合。一个数据库通常包含 n 个表格($n \geqslant 0$)。例如图 10-1 是一张学生基本信息表。

ID	学号	姓名	年龄	班级	性别	籍贯
1	11410104	陈诗颖	20	11物流1班	女	湖南长沙
2	11410207	胡艳	19	11物流2班	女	湖北宜昌
3	09040218	雷亮	22	09经济学2	男	湖北襄阳
4	12360219	顿婷丽	18	12电工2班	女	上海

字段名　记录　值　字段

图 10-1　学生基本情况表

数据库中的一些基本术语如下:

字段:表中竖的一列叫作一个字段,图中有 7 个字段。字段分为字段名和字段值,"姓名"就是一个字段的字段名,"陈诗颖"是该字段的一个字段值。

记录:表中横的一行叫作一条记录,每条记录可描述一个具体的事物(称为实体)。图中选择了第 2 条记录,也就是"胡艳"的相关信息。

值:纵横交叉的地方叫作值,比如图中第 4 条记录的"籍贯"字段的值为"上海"。

域:值的取值范围称为域,例如性别的取值范围是{男,女}。

表:由横行竖列垂直相交而成。可以分为表头(字段名的集合)和表中数据两部分。

表也可看成是若干条记录组成的集合，因此在数据表中不允许有两条完全相同的记录。

数据库：用来组织和管理表的，一个数据库一般有若干张表，这些表之间可以是相互关联的。数据库不仅提供了存储数据的表，而且还包括视图、索引、存储过程等高级功能。

视图：根据用户的需要，只显示表中部分字段或部分记录（单表查询），或者显示来自多个表中的字段或记录（多表查询）。视图返回给用户的查询结果从形式上看也是一张表，但这张表并没有存放在数据库中，而是在内存中通过关系运算得到的，因此是一张“虚表”。视图又称为查询，是普通用户能看到的数据库。

10.1.2 使用 phpMyAdmin 管理数据库

MySQL 是一个开源软件，没有提供图形操作界面，用户可以通过输入命令行来执行各种数据库操作，但这需要用户记住很多的 MySQL 命令。为此，人们开发了 phpMyAdmin 和 Navicat for MySQL，这些软件提供了 MySQL 的图形化操作界面，可以在图形界面下对 MySQL 数据库进行管理。下面介绍它们的使用。

phpMyAdmin 是一个 B/S 结构的软件，它的最大优势在于：用户将其上传到 Web 服务器的网站目录下，就能管理远程服务器上的 MySQL 数据库了。

1. 创建数据库

在如图 6-9 所示的界面中单击 phpMyAdmin 的链接并输入用户名和密码，即可进入 phpMyAdmin 的主界面（见图 10-2），在右侧窗口的“创建一个新的数据库”中可输入待创建的数据库名，如 guestbook。单击“创建”按钮，就创建了一个名为 guestbook 的空数据库。

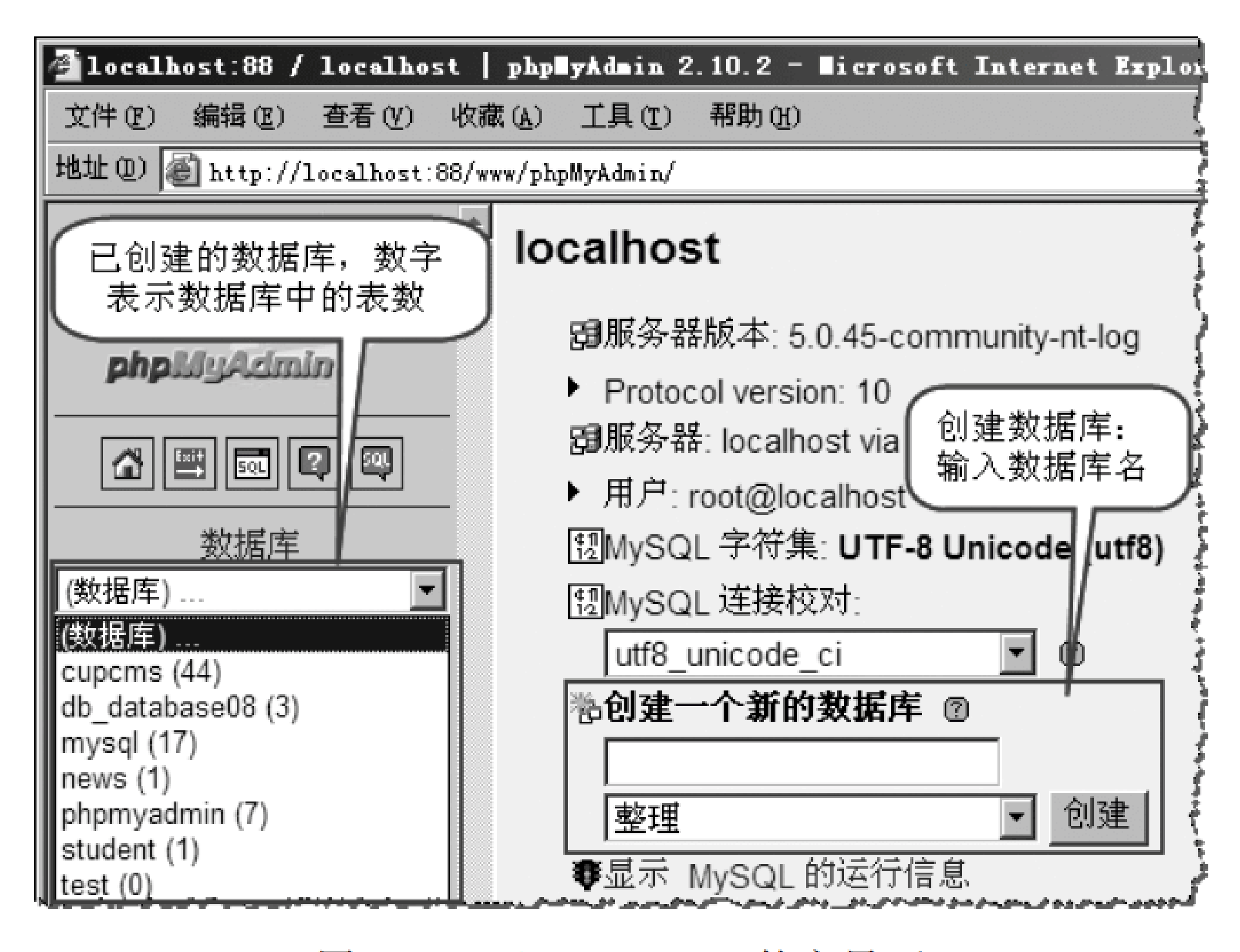


图 10-2 phpMyAdmin 的主界面

提示：创建 guestbook 数据库后，MySQL 会自动在 D:\AppServ\MySQL\data 目录下创建 guestbook 子目录及相关文件（如 db.opt）。因此，一个 MySQL 数据库对应一个目录，如果要移动一个 MySQL 数据库到另一台机器，只需把该数据库对应的目录复制到

另一台机器的\MySQL\data 目录下即可。

2. 新建表

创建数据库后，网页会转到如图 10-3 所示的创建表的窗口。在该窗口中输入待创建表的名称，如 lyb，然后在 Number of fields 后输入表中的字段数，单击“执行”按钮，即创建了一个名为 lyb 的表。此时网页会转到如图 10-4 所示的表的设计视图，在这里需要输入每个字段的名称、类型、长度等信息，并定义主键和额外（如自动编号）等。



图 10-3　phpMyAdmin 创建表的窗口

服务器: localhost ▸ 数据库: guestbook ▸ 表 : lyb

字段	类型	长度/值*1	整理	额外
ID	INT			auto_increment
title	VARCHAR	60	gb2312_chinese_ci	
content	TEXT		gb2312_chinese_ci	
author	VARCHAR	40	gb2312_chinese_ci	
email	VARCHAR	30		
ip	VARCHAR	20		
addtime	DATETIME			
sex	CHAR	2	gb2312_chinese_ci	

设为自动编号字段　设为主键　创建表　保存　或 添加 1 字段　执行

图 10-4　输入表中的字段

由于本节创建的表 lyb 是一个给留言板保存用户留言的表。因此该表中的字段（title、content、author、email、ip、addtime、sex）分别用来保存留言的标题、内容、留言者、留言者的 email、留言者的 IP、留言时间、留言者性别。

图 10-4 中的一行对应一个字段，也就是表中的一列，其中字段名称建议用英文命名，这样方便以后使用 PHP 程序访问表中字段。对每个字段还可以添加注释。MySQL 中，字段的数据类型主要有以下几种。

- INT：用于存储标准的整数，该类型数据占 4 个字节（不能设定长度），取值范围从 −214 783 648～214 783 647。如果要存储的整数比较小，还可考虑使用 tinyint

(取值范围－127～127)和 smallint 数据类型(－32 768～32 767)。

- VARCHAR：是一种可变长度的字符串类型，它可设定长度，其长度范围在 0～255 个字符之间，用于存储比较短的字符串。
- CHAR：是一种固定长度的字符串类型，这种字段占用的空间被固定为创建时所声明的长度。
- TEXT：用于存储比较长的字符串，或图像、声音等二进制数据。该类型不能指定长度。TEXT 和 BLOB 类型在分类和比较上存在区别。BLOB 类型区分字母大小写，而 TEXT 不区分大小写。比指定类型支持的最大范围大的值将被自动截短。
- BOOL：即布尔型数据，它只能取 True 和 False 两个值。
- DATETIME：用于保存日期/时间的数据类型，该类型不能指定长度。此外，如果只希望保存日期，可使用 DATE 类型；如果只要保存时间，可使用 TIME 类型。

提示：CHAR 和 VARCHAR 的区别在于：假设将一个长为 10 字节的字符串保存在一个类型为 CHAR(40)的字段中，则该字符串将占用 40 字节，MySQL 会自动在它的右边用空格字符补足。而如果将其保存在类型为 VARCHAR(40)的字段中，则该字符串只占用 11 字节(每个值只占用刚好够用的字节，再加上一个用来记录其长度的字节)。可见，要节省存储空间，可以使用 varchar 类型，而从速度方面考虑，最好使用 char 类型。

在图 10-4 中，将留言的 ID 字段设置为 auto_increment(自动递增)，这样每插入一条记录，系统都会自动为该记录的 ID 字段添加一个递增的数值，以保证每条记录都会有一个唯一的编号，在查找或显示留言时可以依据这个编号找到对应的留言。留言的内容(content 字段)必须采用 TEXT 数据类型，以保证它可以容纳很长的文本内容。

最后可以对表设置主键，所谓主键，是指能唯一标识某条记录的字段。作为主键的字段必须能满足两个条件：

(1) 该字段中的值不能为空；

(2) 字段中的值不能有重复的，这样该字段才能唯一标识某条记录。

本例中 ID 是自动递增字段，自然不会有重复，也不会有空值，因此可将其作为主键。设置主键的方法是将该字段右侧表示“主键”的单选按钮选中即可。

3. 修改表的结构

创建了表以后，在图 10-3 的左侧窗口，数据库 guestbook 下面就会显示该数据库中已存在的表，如果要修改表的结构，可以在图 10-3 的左侧窗口中单击该表名，右侧窗口就会出现如图 10-5 所示的数据表管理界面，在这里可对数据表的结构进行修改，例如在表中添加字段、删除字段或修改字段。

4. 向表中添加记录

在图 10-5 的数据表管理界面中，单击上方的“插入”选项卡，就会出现如图 10-6 所示的窗口，在这里可以向表中添加记录(即向表中插入一行)，一次最多只能添加两条记录。

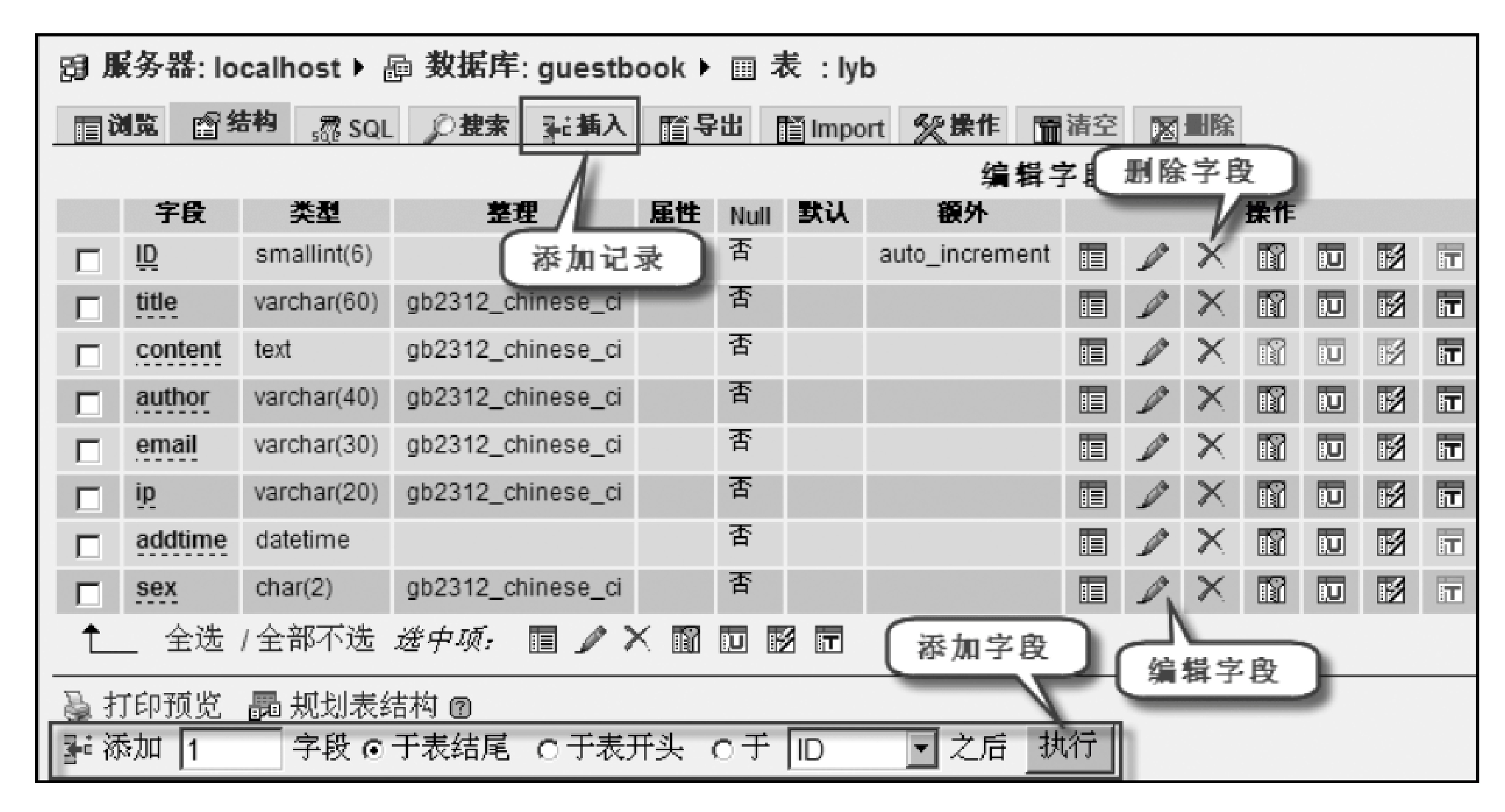


图 10-5　数据表的管理界面



图 10-6　向表中添加记录(插入一行)

说明：

(1) 在各字段输入值时必须符合字段数据类型及该字段格式的要求，否则无法输入；

(2) 不要输入自动编号字段的值，因为系统会自动添加，删除某一记录后自动编号字段(ID)的值也不会被新记录占用。

5. 修改或删除记录

如果要修改或删除一条记录，可以单击图 10-5 中的"浏览"选项卡，则页面下方将显示表中所有记录，如图 10-7 所示。在每条记录的左侧均有"编辑"和"删除"按钮，单击对应按钮即可对该条记录进行编辑或删除。

图 10-7 修改或删除表中的记录

6. 修改表名或复制表

在图 10-5 上方的“操作”选项卡中，可以对表名进行修改，只要在“表选项”中，在“将表改名为”后面输入新表名即可。在右侧的“将表复制到”选项中，还可将表复制到其他数据库中，或复制为本数据库中其他的表。

10.1.3 使用 phpMyAdmin 导出导入数据

1. MySQL 数据库的迁移

有时为了把 PHP 程序迁移到另一台计算机上运行，或上传到服务器上。就需要将 PHP 程序访问的 MySQL 数据库也一起迁移。迁移 MySQL 数据库有两种方案：

(1) 复制目录的方式。每个 MySQL 数据库对应一个目录，例如数据库 lyb 对应 D:\AppServ\MySQL\data\lyb 目录，该目录下保存了数据库中的相关数据。如果要移动数据库 lyb 到另一台机器，只需把该数据库对应的目录复制到另一台机器的\MySQL\data 目录下即可(复制之前最好先停止 MySQL 服务器)。

(2) 导出和导入.sql 文件的方式：可以将一个数据库(或表)导出成一个.sql 文件，该.sql 文件中包含了很多条 SQL 命令，用来创建所有表的结构和插入数据。在另一台机器上先创建一个空数据库，再将这个.sql 文件导入到空数据库中即可。

2. 使用 phpMyAdmin 导出和导入数据

(1) 在 phpMyAdmin 中导出数据库的步骤是：在图 10-3 的左侧单击选择要导出的数据库(或表)，然后单击图 10-3 上方的“导出”选项卡，就会出现“数据库的转存”界面(见图 10-8)，选中“另存为文件”复选框，单击“执行”按钮，就会生成并提示下载“*.sql”的文件。

(2) 导入数据库的步骤是：首先创建一个空数据库(见图 10-2)，然后单击图 10-3 上方的 Import 选项卡，就会出现导入 sql 文件界面(见图 10-9)，选择一个.sql 的文件，然后指定该文件的字符集(若不知道文件的字符集，可以用记事本打开该文件，然后选择“文件”→“另存为”命令，在“另存为”对话框下方的“编码”中，就可以看到它的字符集)，单击“执行”按钮，如果提示导入成功，就将数据导入进了空数据库中。

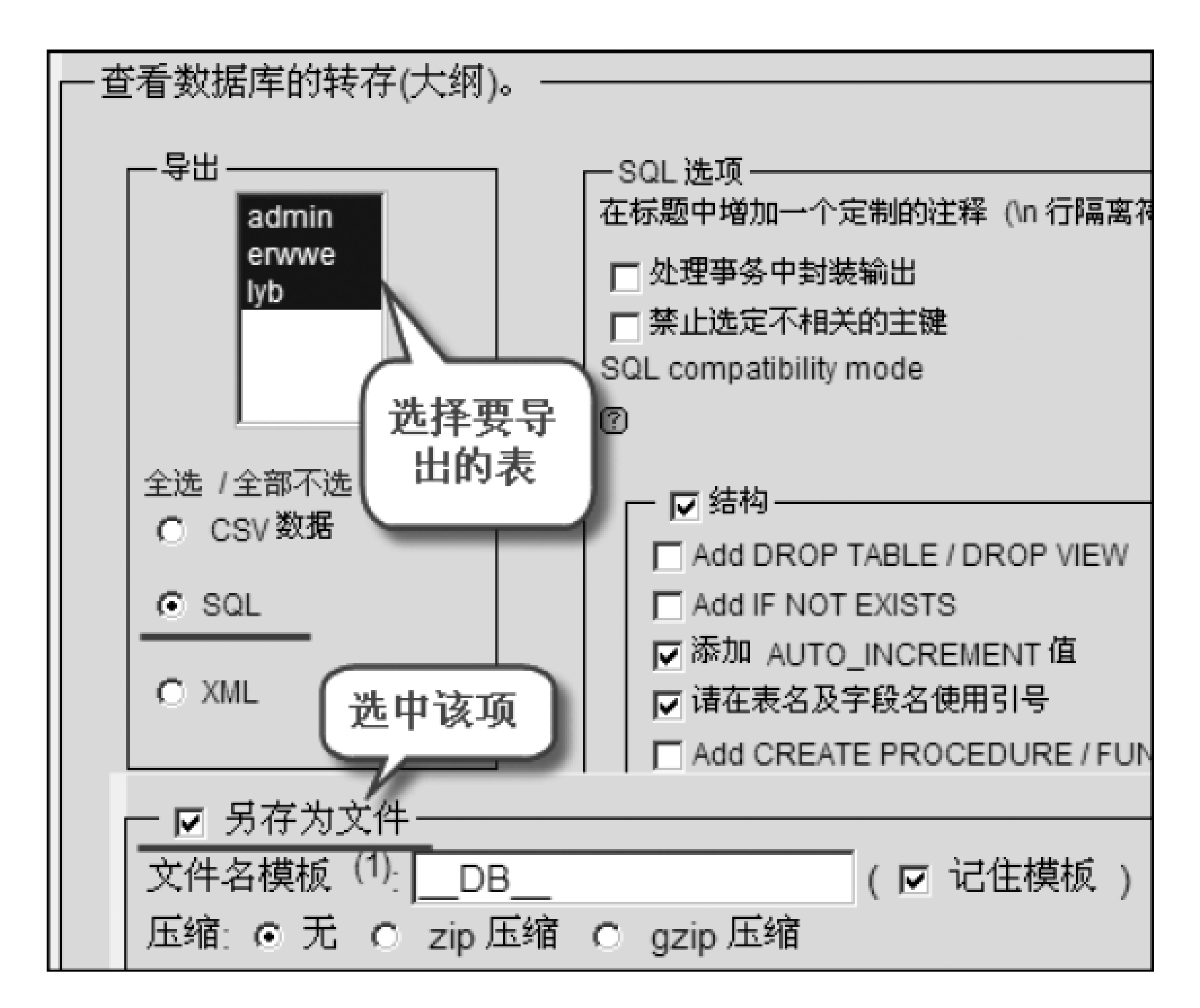


图 10-8　数据库的转存



图 10-9　导入.sql 文件

10.1.4　使用 Navicat 管理数据库

Navicat for MySQL 是一个 C/S 结构的 MySQL 数据库管理系统，其功能比 phpMyAdmin 更加强大。Navicat 10.1 的界面如图 10-10 所示，下面来讲解它的使用方法。



图 10-10　Navicat for MySQL 的界面

1. 连接 MySQL 服务器

Navicat 在使用前，必须先连接数据库服务器，单击图 10-10 工具栏中的“连接”按钮，

将弹出"新建连接"对话框(见图 10-11),连接名可任取一个名字,输入正确的用户名和密码即可。连接成功后,在图 10-10 左侧的导航窗口中双击连接图标 conn,就可看到本机上所有的 MySQL 数据库。

图 10-11 "新建连接"对话框

2. 新建数据库

在图 10-10 的 conn 或数据库图标上右击,选择右键菜单中的"新建数据库",然后输入数据库名即新建了一个数据库。

3. 新建表

新建完数据库后,接下来当然是新建表了。在图 10-10 左侧导航栏中的"表"上右击,选择"新建表"命令,将弹出新建表窗口(见图 10-12),在这里可定义表中的每个字段,并设置主键等。其中"栏位"表示"字段",单击"添加栏位"就可添加一个字段。定义完字段后,单击"保存"按钮,输入表名即新建了一个表。如果以后要修改表的结构,在图 10-10 中单击该表单击工具栏下方的"设计表"按钮就可以了。



图 10-12 新建表窗口

4. 在表中添加记录

在图 10-10 的右侧窗口双击某个表(如 news),将弹出表的数据视图(见图 10-13),单击"+"按钮,就可以插入一条记录。单击"-"按钮,可删除一条记录。



图 10-13 表的数据视图

5. 导出数据库

在图 10-10 中,在某个数据库(如 abced)上右击,选择"转储 SQL 文件"命令,单击"保存"按钮,即可将数据库导出成.sql 文件。

6. 导入数据库

首先新建一个空数据库,然后在图 10-10 左侧的数据库图标上右击,选择"运行 SQL 文件"命令,单击"…"按钮,选择一个要导入的.sql 文件,单击"开始"按钮,则开始执行该

. sql 文件中的代码,如果执行成功,就会将所有表及数据导入到该数据库中来。

7. 将 Access 数据库转换成 MySQL 数据库

首先新建一个数据库(如 lyb),然后在图 10-10 中双击展开数据库,在“表”上右击,选择“导入向导”命令,就会弹出如图 10-14 所示的“导入向导”对话框,选择“MS Access 数据库”单选按钮。单击“下一步”按钮,选择一个. mdb Access 数据库文件,在下方将出现该数据库中所有的表,选中要转换的表,单击“下一步”按钮,可设置是否新建目标表及目标表的名称,均保持默认即可,继续单击“下一步”按钮,即可将 Access 数据库转换成 MySQL 数据库。

提示:*使用 Access2MySQL Pro 5 软件也可将 Access 数据库转换成 MySQL 数据库。*

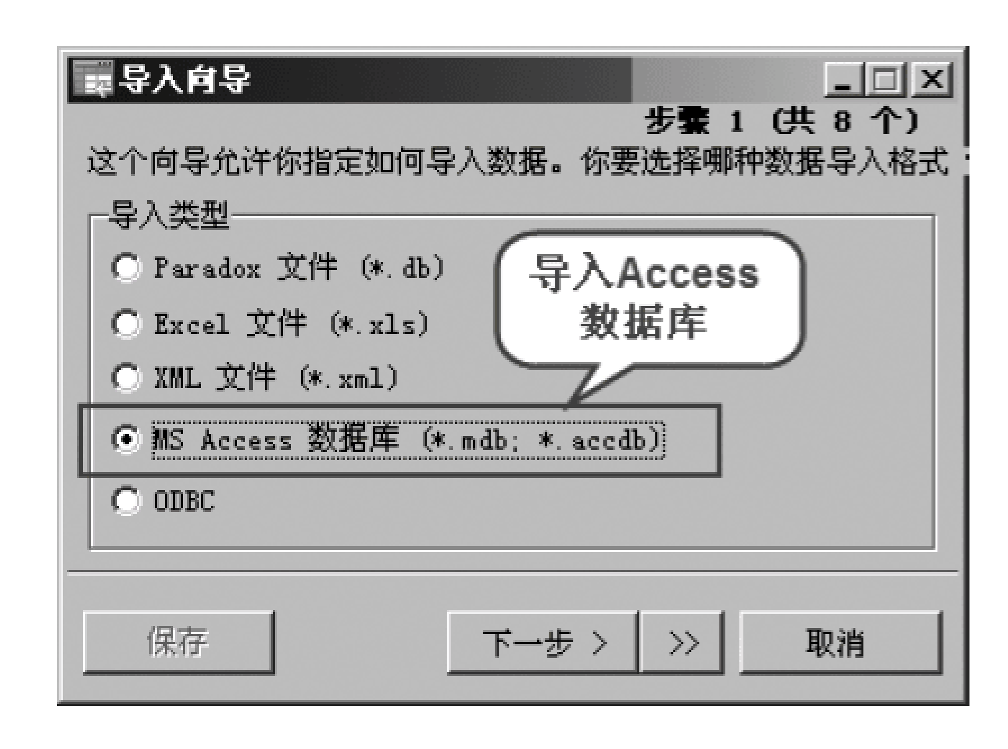


图 10-14　“导入向导”对话框

10.2　SQL 语言

SQL(Structured Query Language)即结构化查询语言,是操作各种数据库的通用语言。在 PHP 中,无论访问哪种数据库,都需要使用 SQL。SQL 语言本身是比较庞大复杂的,但制作普通动态网站只需掌握以下最常用的 SQL 语句就够了。

(1) Select 语句——查询记录,基本形式为 Select…from…。

(2) Insert 语句——添加记录,基本形式为 Insert into…values…。

(3) Delete 语句——删除记录,基本形式为 Delete from…[where]…。

(4) Update 语句——更新记录,基本形式为 Update…Set…。

(5) Create 语句——创建表或数据库,基本形式为 Create table(或 Database)…。

10.2.1　Select 语句

Select 语句用来实现对数据库的查询。简单地说,就是可以从数据库的相关表中查询符合特定条件的记录(行)或字段(列)。语法如下:

```
Select 字段列表 From 表 [Where 条件] [Order By 字段] [Group By 字段] [limit s, n]
```

说明:

(1) 字段列表:即要显示的字段,可以是一个或多个字段名,多个字段之间用逗号隔开。用“*”表示全部字段。

(2) 表:指要查询的数据表的名称,如果有多个表,则中间用逗号隔开。

(3) Where 条件:就是查询只返回满足这些条件的记录。

(4) Order By 字段:表示将查询得到的所有记录按某个字段进行排序。

(5) Group By 字段:表示按字段对记录进行分组统计。

(6) limit s, n: 表示选取从第 s 条记录开始的 n 条记录,如果省略 s,则表示选取前 n 条记录。如选取前 6 条记录,就是 limit 6。

1. 常用的 Select 语句示例

(1) 选取数据表中的全部数据(所有行所有列):

```
Select * from lyb
```

(2) 选取指定字段的数据(即选取表中的某几列):

```
Select author, title from lyb
```

(3) 选取数据表中最前面几行或中间几行记录

```
Select * from lyb limit 5          //选取前 5条记录 (limit n等价于 limit 0, n)
Select * from lyb limit 0, 5       //选取前 5条记录 (记录行号从 0开始)
Select * from lyb limit 5, 10      //选取第 6到 15条记录
```

说明:MySQL 或 Oracle 等数据库使用 limit 关键字来限制返回的结果集。limit 只能放置在 Select 语句的最后位置。语法为:

```
limit [首行行号,] 记录条数
```

对于微软的数据库产品,要选取表中前 n 条记录,必须使用 Select top n 的语法。例如:

```
Select top 5 * from lyb            //选取前 5条记录,非 MySQL数据库的写法
```

(4) 选取满足条件的记录

```
Select * from lyb where ID> 5
Select * from lyb where author= '张三'
Select author, title from lyb where ID Between 2 And 5     //如果条件是连续值
Select * from lyb where ID in(1, 3, 5)                     //如果条件是枚举值
```

由此可见,Select 子句用于从表中选择列(字段),Where 子句用来选择行(记录)。

说明:

(1) 在 SQL 语句中用到常量时,字符串常量两边要加单引号(如'张三'),日期和时间两边要加#号,而数值常量可加单引号,也可不加(如 5)。

(2) SQL 语言不区分大小写,但在 PHP 中书写 SQL 语句,字段名是区分字母大小写的。

2. 选取满足模糊条件的记录

有时经常需要按关键字进行模糊查询,例如,

```
Select * From lyb Where author like '%芬%'     //author字段中有"芬"字的记录
Select * From lyb Where author like '张%'      //姓名以张开头的人
```

```
Select * From lyb Where author like '唐_'        //姓名以唐开头且为单名的人
```

其中,“%”表示与任何 0 个或多个字符匹配,“_”表示与任何单个字符匹配。需要注意的是,如果在 Access 中直接写查询语句,“%”需换成“*”,“_”需换成“?”。

3. 对查询结果进行排序

利用 Order By 子句可以将查询结果按照某种顺序排序出来。例如,

```
Select * From lyb order by author ASC
Select * From lyb order by id desc          //根据 id字段降序排列
```

如果要按多个字段排序,则字段间用逗号隔开。排序时,首先参考第一字段的值,当第一字段值相同时,再参考第二字段的值,以此类推。例如:

```
Select * From lyb order by date DESC, author
```

说明:ASC 表示按升序排列,DESC 表示按降序排列。如果省略,默认值为 ASC。

4. 汇总查询

有时需要对全部或多条记录进行统计。比如对一个学生成绩表来说,可能希望求某门课程所有学生的平均分。又如对学生信息表来说,可能需要求每个专业的学生人数。Select 语句中提供了 Count、Avg、Sum、Max 和 Min 共 5 个聚合函数,分别用来求记录总数、平均值、和、最大值和最小值。下面是两个汇总查询的示例。

```
Select count(*) From lyb                    //查询 lyb表中的记录总数
Select avg(id),sum(id),max(id) From lyb
```

说明:

(1) 以上例子返回的查询结果都只有一条记录,即汇总值。

(2) Count(*)表示对所有记录计数。如果将*换成某个字段名,则只对该字段中非空值的记录计数。

(3) 如果在以上例子中加上 Where 子句,将只返回符合条件的记录的汇总值。

聚合函数还可以与 group by 子句结合使用,以便实现分类统计。比如要统计每个系的男生人数和女生人数的 Select 语句如下:

```
Select 系名, sex, count(*) From students Group By 系名, sex
```

注意:使用 group by 子句时,Select 子句中只能含有 group by 中出现过的字段名。

5. 多表查询

如果要查询的内容来自多个表,就需要对多个表进行连接后再进行查询。

例如,某购物网站的数据库中含有两个表:商品表(goods)和购物车表(cart)。商品表中包含了商品 id、商品名、商品图片、型号、单价、商品描述等字段。购物车表中包含了用户 id、商品 id、商品数量等字段。两个表的结构如下:

商品表：goods(spID, Name, Picture, Type, Price, descrpt)

购物车表：cart(UserID, spID, Number)

但一般的购物车网页中，往往还需要将商品的图片、名称、单价等信息显示出来，如图 10-15 所示，以便顾客能清楚地看到购物车中的各种商品。但购物车表中却只保存了商品 id(spID)，并没有保存商品的其他属性。为此，可以通过商品 id 字段(两个表中共有的字段)，将购物车表和商品表连接起来，就能查询到商品名称、图片、单价、用户 id 和数量等存储在两个表中的信息了。Select 语句如下：

```
Select Name, Picture, Price, Number, Number * Price from goods, cart where goods.spID= cart.spID and
cart.userID= 'tangsix'
```



图 10-15　一个顾客购物车网页

说明：

(1) 在多表查询中，如果若干个表中都有同一个字段名，则字段名必须写成"表名.字段名"，以指定该字段是某个表的，如 cart.spID 表示 cart 表的 spID 字段。

(2) 上述查询的 where 子句中的 goods.spID=cart.spID 表示将两个表通过 spID 进行连接，如果是多表查询，那么这样的连接条件一定不能省略。

6. 其他辅助查询功能

使用 distinct 关键字可以去掉查询结果中重复的记录，使用 as 关键字可以为字段名指定别名。例如：

```
Select distinct author From lyb          //多条记录中有相同的作者名则只显示一条
Select author as 作者, title as 标题 From lyb          //将字段名 author 显示为作者
```

10.2.2　添加、删除、更新记录的语句

在 SQL 中，添加、删除、更新记录可分别使用 insert、delete 和 update 语句实现。

1. insert 语句

在动态网站程序中,经常需要向数据库中插入记录。例如用户发表一条留言,就需要将这条留言作为一条新的记录插入到表 lyb 中。使用 insert 语句可以实现该功能,语法如下:

```
Insert Into 表 (字段 1, 字段 2, …) Values(字段 1的值, 字段 2的值, …)
```

说明:

(1) 利用 Insert 语句可以给表中部分或全部字段赋值。Value 括号中的字段值的顺序必须和前面括号中的字段一一对应。各字段之间、字段值之间用逗号隔开。

(2) 在插入记录时要注意字段的数据类型,若为字符串类型,则该字段值的两边要加单引号;若为日期/时间型也应在值两边加单引号,若为布尔型,则值应为 True 或 False;自动递增字段不需要插入值。

(3) 可以只给部分字段赋值,但主键字段必须赋值,不能为空且不能重复。

下面是一些插入记录的例子:

```
Insert Into lyb(author) values('芬芬')
Insert Into lyb(author, title, `date`) values('芬芬','大家好!', '205-12-12')
```

说明: 由于 date 是 SQL 语言中的一个关键字,如果表中的字段名与 SQL 中的关键字相同,就必须把该字段名写在反引号内,如`date`,否则 SQL 语句会出错。因此有时在执行 Insert 语句出现不明原因的错误时,不妨把所有字段名都写在反引号内。

2. Delete 语句

使用 Delete 语句可以一次性删除表中的一条或多条记录。语法如下:

```
Delete From 表 [Where 条件]
```

说明: "Where 条件"与 Select 语句中的 Where 子句作用是一样的,都用来筛选记录。在 Delete 语句中,凡是符合条件的记录都会被删除;如果没有符合条件的记录则不删除;如果省略 Where 子句,则会将表中所有的记录全部删除。

下面是一些删除记录的例子:

```
Delete from lyb where id=17
Delete from lyb where author='芬芬'
Delete from lyb where date<'2010-9-1'
```

提示: Delete 语句以删除一整条记录为单位,它不能删除记录中某个或多个字段的值,因此 Delete 与 from 之间没有 * 或字段名。如果要删除某些字段的值,可以用下面的 Update 语句将这些字段的值设置为空。

3. Update 语句

Update 语句用来修改表中符合条件的记录。语法如下:

```
Update 表 Set 字段 1=字段值 1,字段 2=字段值 2,… [Where 条件]
```

说明：Update 语句可以更新全部或部分记录。其中“Where 条件”是用来指定更新数据的范围，其用法同 Delete 语句。凡是符合条件的记录都会被更新，如果省略条件，则将更新表中所有的记录。

下面是一些常见的例子：

```
Update lyb set email='fengf@163.com' where author='芬芬'
Update lyb set title='此留言已被删除', content=Null where id=16
```

修改记录时，也可以采取先删除再添加记录。不过，这样实际上是添加了一条记录，记录的自动递增值会改变，而有时是需要通过自动递增值来查找记录的，而且采取先删除再添加需要执行两条 SQL 语句，有时可能会发生第 1 条执行成功，第 2 条执行失败的情况，从而对数据产生破坏。

10.2.3 SQL 字符串中含有变量的书写方法

在 PHP 中，如果要执行 SQL 语句，通常将 SQL 语句写在一个字符串中。对于下面的 SQL 语句：

```
Select * from link where name='搜狐'
```

如果要把它写成字符串的形式，则形式如下（因为字符串常量要写在引号中）：

```
$str="Select * from link where name='搜狐'";
```

这样就能从 link 表中查询到网站名 name 是“搜狐”的记录信息。但实际查询时，查询条件（如此处的“搜狐”）通常是从表单中获取的，如 $webName=$_POST['webname']。这样，查询条件就保存到了字符串变量 $webName 中了。

1. SQL 中嵌入字符串变量的情况

字符串常量和变量之间可以使用连接符(.)连接在一起，因此，上面的语句可改为：

```
$str="select * from link where name='".$webName ."'";
```

在这条语句等号右边的表达式中，实际包括如下三部分内容，即两个字符串常量和一个字符串变量，它们之间用连接符“.”连接在一起：

第一部分，字符串常量："select * from link where name='"。

第二部分，字符串变量：$webName。

第三部分，字符串常量："'"。

这几部分容易引起迷惑的是，为什么第一部分和第三部分中既有单引号又有双引号呢？其实，两边的双引号就是表示中间的内容是一个字符串常量。其中的单引号'和别的字符（如 abcd）一样，只是这个字符串常量的内容而已。同样，对于第三部分来说，两边的双引号表示这是一个字符串常量，中间的单引号就是它的内容。

而 PHP 提供了双引号字符串，这种字符串中可包含变量，因此，上面的语句又可

写为：

```
$str="select * from link where name='$webName '"
```

2. SQL 中嵌入数值型变量的情况

如果 SQL 语句中的常量是数值型，则可以用单引号括起来(从安全性方面考虑，推荐这样做)，也可以不用单引号括起来。例如：

```
Select * from link where ID=5
```

把它写成 SQL 字符串就是：

```
$str="Select * from link where ID=5";
```

如果变量 $ linkid=5，则将语句中的数值 5 替换成数值型变量 $ linkid 后的字符串如下：

```
$str="Select * from link where ID=".$linkid;
```

可见它由两部分组成，即前面的"Select * from link where ID="是字符串常量，后面的 linkid 是数值型变量，它们之间用连接符“.”连接起来。

3. SQL 语句中含有多个变量的情况

在 SQL 语句(尤其是 Insert 语句)中，经常会碰到一条 SQL 语句中有多个变量的情况，对于下面保存在 $ str 中的 Insert 语句：

```
$str="Insert Into lyb(author, title) values('芬芬','大家好!') ";
```

如果变量 $ user='芬芬', $ tit='大家好!'，则可将该 SQL 字符串改写为：

```
$str="Insert Into lyb(author, title) values('" .$user . "','".$tit . "')";
```

可见它由五部分组成，分别是字符串常量"Insert Into lyb(author,title) Values('"、字符串变量 $ user、字符串常量"','"、字符串变量 $ tit 和字符串常量"')"，它们通过四个“.”连接起来。

10.3　访问 MySQL 数据库

动态网站、Web 应用程序都需要数据库的支持。将网站数据库化，就是使用数据库来管理整个网站。这样只需更新网站数据库的内容，网站页面内容就会自动更新。网站数据库化的好处有：

(1) 可以自动更新网页。采用数据库管理，只要更新数据库的数据，网页内容就会自动得到更新，过期的网页也可以自动不显示。

(2) 加强搜索功能。将网站的内容存储在数据库中，可以利用数据库提供的强大搜索功能，从多个方面搜索网站内的信息。

（3）可以实现各种基于 Web 数据库的应用。用户只要使用浏览器，就可以通过网络，查询或存取位于 Web 服务器数据库中的数据，实现 Internet 上的各种应用功能。例如个人博客、网上购物、网上订票、银行余额查询、股市买卖交易以及网上择友等。

因此，很多人认为动态网站就是使用了数据库技术的网站，这种说法虽不准确，但足以说明数据库在动态网站中的扮演着重要角色。

10.3.1 PHP 访问数据库的步骤

PHP 之所以最适合与 MySQL 数据库搭配使用，主要原因是 PHP 提供了很多操作 MySQL 数据库的内置函数，这些函数可方便地实现访问和操纵 MySQL 数据库的各种需要。PHP 访问 MySQL 数据库有三种方式：

（1）使用 MySQL 内置函数；

（2）使用 Mysqli 内置函数；

（3）使用 PDO 数据接口层。

用 PHP 访问 MySQL 数据库的一般步骤如下（见图 10-16）：

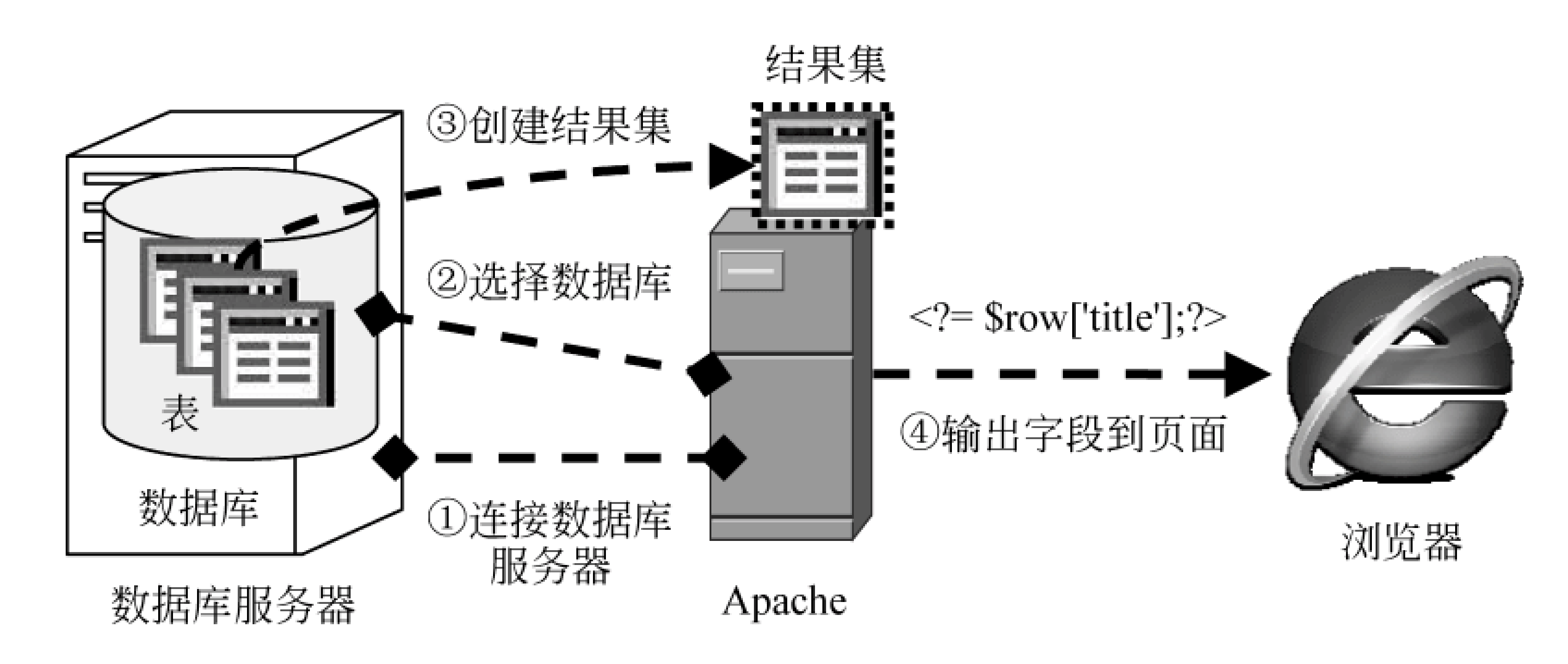


图 10-16　PHP 访问 MySQL 数据库的一般步骤

（1）连接数据库服务器并选择数据库，这样 PHP 程序就与数据库建立了连接；

（2）创建结果集。即通过执行查询语句将数据表中符合条件的行读取到服务器内存中，此时内存中保存了查询得到的“虚表”，就称为结果集。这样 PHP 就获得了访问表中数据的能力。

（3）绑定数据到页面。即输出结果集中某条记录中一个或多个字段的值到页面上。PHP 通常是将一条记录存储到一个数组中，再输出字段对应的数组元素实现的。

通过以上三步，网页上就可以显示数据库表中的数据了，如图 10-17 所示。



图 10-17　使用浏览器显示数据库中的数据

PHP 访问 MySQL 数据库的具体步骤是：

(1) 建立与 MySQL 服务器的连接；

(2) 设置字符集；

(3) 选择要操作的数据库；

(4) 创建结果集；

(5) 将结果集中的记录读入数组中；

(6) 在网页上输出数组元素的值。

10.3.2 连接 MySQL 数据库

要连接 MySQL 数据库，需要先连接 MySQL 服务器，然后设置数据库的字符集，再选择数据库，就可以和指定的数据库建立连接了。

1. 连接 MySQL 服务器

进行 MySQL 数据库操作之前，首先要与 MySQL 服务器建立连接。PHP 中连接 MySQL 服务器的函数是 mysql_connect()。该函数语法如下：

```
resource mysql_connect(string hostname, string username, string password)
```

函数功能是：通过 PHP 程序连接 MySQL 数据库服务器，如果连接成功，则返回一个资源类型的 MySQL 服务器连接标识(link_identifier)，否则返回 false。例如：

```
$conn=mysql_connect("localhost","root","111");
```

表示连接主机名为 localhost 的数据库服务器，其用户名为 root，密码是 111。

提示：连接 MySQL 服务器的过程需要耗费大量的服务器资源，为了提高系统性能及资源利用率，如果在同一个脚本中多次连接同一个 MySQL 服务器，PHP 将不会创建多个 MySQL 服务器连接，而是使用同一个 MySQL 服务器连接。

2. 设置数据库字符集

PHP 与 MySQL 进行信息交互之前，为了防止中文乱码，必须用 mysql_query()方法将数据库字符集设置为与网页相同的字符集，例如，网页的字符集是 gb2312，就必须将数据库的字符集也设置为 gb2312。设置字符集的代码如下：

```
mysql_query("set names 'gb2312'");
```

提示：在 Dreamweaver 8 版本中新建的网页字符集默认是 gb2312，而 Dreamweaver CS3 以上版本中新建的网页字符集默认是 utf-8，在 Dreamweaver 中，执行“修改→页面属性”菜单命令，找到“标题/编码”选项卡，在“编码”下拉列表框中就可修改网页的字符集。

3. 选择数据库

由于 MySQL 服务器中可以有多个数据库，为了指定要访问的数据库，需要使用 mysql_select_db()方法设置当前操作的数据库。例如，选择当前数据库为 guestbook，则

代码如下：

```
mysql_select_db("guestbook",$ conn);
```

对于一个动态网站来说，几乎所有的页面都要连接同一个数据库。为此，可将连接数据库的代码单独写在一个文件中（该文件一般命名为 conn.php），在需要连接数据库的网页中使用 require 或 include 命令包含它即可。代码如下：

```
---------------------清单 10-1 conn.php------------------
<?
$conn=mysql_connect("localhost","root","111");        //连接数据库服务器
mysql_query("set names 'gb2312'");                    //设置字符集
mysql_select_db("guestbook",$conn);                   //选择数据库
?>
```

10.3.3 创建结果集并输出记录

连接了数据库以后，PHP 程序只是和指定的数据库建立了连接，但数据库中通常有多个表，数据库中的数据都是存储在表中的。为了在页面上显示数据，必须读取指定的表（全部或部分数据）到内存中来，这称为创建结果集（result）。结果集可以看成是内存中的一个虚表，由若干行或若干列组成。结果集带有一个记录指针，在刚打开结果集时指针指向结果集中第一条记录（若结果集不为空），如图 10-18 所示。



图 10-18　结果集示意图

使用 mysql_query()方法可以向 MySQL 服务器发送一条 Select 语句，此时该函数将返回一个结果集（result）。代码如下：

```
$result=mysql_query("select * from lyb",$conn);     //创建结果集
```

1. 在页面上输出整条记录

结果集相当于内存中的一个表。可以使用 mysql_fetch_assoc()等函数读取结果集中的一行到数组中。mysql_fetch_assoc()函数的参数是一个结果集，返回值是一个数组，该数组中保存了结果集指针当前指向的行；如果结果集指针没有指向行，则返回 false。

然后就可以输出数组元素到网页中，这样网页上就能显示数据表中的一条记录了。代码如下：

```
$row=mysql_fetch_assoc($result);                    //取出结果集中当前指针指向的行并保存到数组
echo $row['title'].' '.$row['author'].' '.$row['email'];      //输出数组元素
```

输出结果为：

```
祝大家开心 唐三彩 sanyo@tom.com
```

其中，$ row=mysql_fetch_assoc($ result)的作用是将结果集指针当前指向的记录保存到数组 $ row 中，然后将结果集指针下移一条记录，如图 10-19 所示。

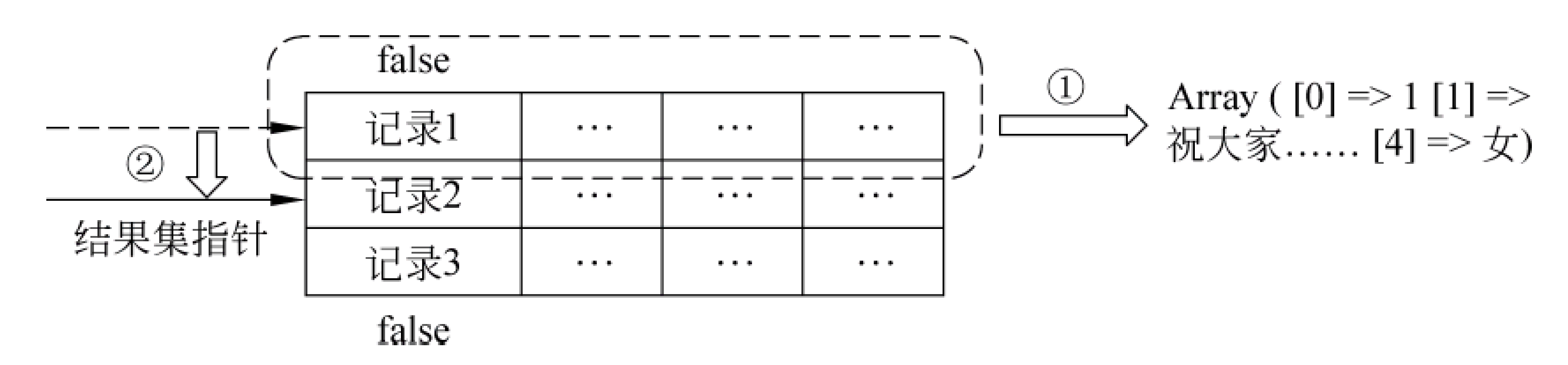


图 10-19　mysql_fetch_ * 函数的功能

实际上，PHP 提供了 4 个形如 mysql_fetch_ * ()的函数，都可以读取结果集中的当前记录到数组中，并将结果集指针移动到下一条记录。这 4 个函数分别是：

(1) mysql_fetch_row()：将当前记录保存到一个索引数组中。

(2) mysql_fetch_assoc()：将当前记录保存到一个关联数组中。

(3) mysql_fetch_array()：将当前记录保存到一个含有索引和关联的混合数组中。

(4) mysql_fetch_object()：将当前记录保存到一个对象中。

前 3 个函数的区别仅仅在于保存记录的数组不同。例如，假设已创建了一个结果集 $ result，则可以分别用这 3 个函数读取结果集中当前记录到数组中，再打印数组出来。示例如下：

(1) 输出 mysql_fetch_row()保存记录的数组。

```
$row=mysql_fetch_row($result);
print_r($row);
```

输出结果为：

```
Array([0]=>1 [1]=>祝大家开心 [2]=>非常感谢大家的帮助 [3]=>唐三彩 [4]=>sanyo@tom.com [5]=
>59.51.24.37 [6]=>2012-03-20 00:00:00 [7]=>女)
```

(2) 输出 mysql_fetch_ assoc()保存记录的数组。

```
$row=mysql_fetch_assoc($result);
print_r($row);
```

输出结果为：

```
Array([ID]=>1 [title]=>祝大家开心 [content]=>非常感谢大家的帮助 [author]=>唐三彩 [email]=
>sanyo@tom.com [ip]=>59.51.24.37 [date]=>2012-03-20 00:00:00 [sex]=>女)
```

(3) 输出 mysql_fetch_ array()保存记录的数组。

```
$row=mysql_fetch_array($result);
print_r($row);
```

输出结果为：

```
Array([0]=>1 [ID]=>1 [1]=>祝大家开心 [title]=>祝大家开心 [2]=>非常感谢大家的帮助
[content]=>非常感谢大家的帮助 [3]=>唐三彩 [author]=>唐三彩 [4]=>sanyo@tom.com [email]=>
sanyo@tom.com [5]=>59.51.24.37 [ip]=>59.51.24.37 [6]=>2012-03-20 00:00:00 [date]=>2012-03-20 00:
00:00 [7]=>女 [sex]=>女)
```

可见，mysql_fetch_row()返回的数组索引是数字，mysql_fetch_ assoc()的数组索引是字符串(表的字段名)，而 mysql_fetch_array()的数组内容实际上是上面两个函数数组的合集，它同时保存了两种数组元素。

由于在实际开发中一般不知道要输出字段的序号，但知道字段的字段名，因此 mysql_fetch_assoc()比 mysql_fetch_row()更常用。而 mysql_fetch_array()由于生成的数组元素太多，占用内存资源，建议少用。

(4) 输出 mysql_fetch_object()保存的记录。

mysql_fetch_object()返回一个对象，该对象的各个属性中保存了当前记录中各个字段的值。通过“对象->字段名”可返回当前记录中某个字段的值。例如：

```
$row=mysql_fetch_object($result);          //将当前记录保存到对象$row中
echo $row->title;                          //输出 title字段的值,如“祝大家开心”
```

2. 在页面上输出单个字段

创建了结果集后，使用 mysql_result()函数可以返回结果集指针当前指向记录的某个字段值。该函数语法为：mysql_result(result, row, field)，其中，result 为一个结果集资源，row 用来指定行号(行号从 0 开始)，field 是字段名或字段序号。例如：

```
echo mysql_result($result,1,'author');      //输出"小神马"
```

就可以输出结果集中第二条记录的 author 字段的值。输出完之后，mysql_result()函数也会将结果集指针移动到下一条记录。

3. 通过循环输出所有记录

如果要输出结果集中的所有记录，可以将 mysql_fetch_ assoc()放在循环语句中执行，这样第一次循环时将取出第一条记录到数组中，然后结果集指针下移一条记录。第二次循环时将取出指针指向的第二条记录，结果集指针又下移一条记录。如此循环，直到结果集指针指向了结果集的末尾(最后一条记录之后)才停止循环。但是这样只能输出每条记录的内容(每个字段值)。如果要以表格的形式输出结果集，则必须用 HTML 标记定义表格，再将结果集中的字段值输出到每个单元格<td>中。

下面是一个将表 lyb 中数据显示在页面上的完整程序。其运行结果如图 10-20 所示。

```
-----------------------清单 10-2.php 显示数据库中的记录-----------------
<?                             /* 连接数据库 */
$conn=mysql_connect("localhost","root","111");        //连接数据库服务器
```

```
mysql_query("set names 'gb2312'");                    //设置字符集
mysql_select_db("guestbook",$conn);                   //选择数据库
$result=mysql_query("Select * from lyb",$conn);       //创建结果集
?>
<!--------------在页面上显示数据库中的记录---------------------->
<table border="1" width="95%">
  <tr bgcolor="#e0e0e0">
    <th>标题</th><th width="100">内容</th><th width="60">作者</th>
    <th>email</th><th width="80">来自</th></tr>
  <?    //循环输出记录到页面上
    while($row=mysql_fetch_assoc($result)){     ?>
  <tr><td><?=$row['title']?></td><td><?=$row['content']?></td>
    <td><?=$row['author']?></td><td><?=$row['email']?></td>
    <td><?=$row['ip']?></td></tr>
  <?}?>
</table>
```



标题	内容	作者	email	来自
祝大家开心	非常感谢大家的帮助	胡积分	xta@tom.com	202.103.56.6
请教个问题	虚拟目录中的文件如何预览	唐三彩	renw@qq.com	127.0.0.1
这是测试留言	学习ASP程序设计的过程真是其乐无穷	王承芬	tang@163.COM	192.168.0.1
第五条留言	在古巴比伦的大草原上	喻志	yuxh@sohu.com	127.0.0.1

图 10-20　程序 10-2.php 的运行结果

说明：

（1）本程序分为三部分：第一部分是连接数据库服务器和选择数据库；第二部分是利用 mysql_query()方法创建结果集；第三部分是用 while 循环读取结果集中的所有记录。

（2）刚打开结果集时，指针指向第一条记录，执行 mysql_fetch_assoc($result)方法会先将这条记录赋给数组$row，然后使指针移到下一条记录，这样第二次循环时将输出第二条记录。当指针移动到最后一条记录之后时，该方法将返回 false，这样$row 的值也变成 false，循环将不再继续。

想一想：

```
<? while($row=mysql_fetch_assoc($result)){     ?>
```

能否改为：

```
<? while(mysql_fetch_assoc($result)){$row=mysql_fetch_assoc($result);     ?>
```

（3）由于每次循环显示一条记录，而每条记录显示在一行中，因此 while 循环的循环体是一对<tr>…</tr>标记中的内容。

(4) 字段名是区分大小写的。假如将 $ row['title']写成 $ row['Title'],则该字段值无法输出。

提示: 从该程序可以看出,PHP 程序无法用一条语句将结果集(整个表)按原样输出,而只能利用循环将结果集中记录一条一条地输出。

4. 输出指定的 *n* 条记录

如果不想输出所有记录,只想输出结果集中的前 *n* 条记录,那么至少有两种方法,一种是使用 for 循环,限定循环次数为 *n*;第二种方法是修改 SQL 语句为 Select * from lyb limit n,这样结果集中就只有 *n* 条记录。推荐用第二种方法,因为前一种方法虽然只在页面上输出 *n* 条记录,但实际上已经把所有的记录都读取到了结果集中,占用了内存。

5. 返回记录总数 mysql_num_rows()

该函数可以返回结果集中的记录总数,其参数是一个结果集资源。例如:

```
<p>共有<?=mysql_num_rows($result) ?>条记录</p>
```

6. mysql_db_query()函数

mysql_db_query()函数可以同时选择数据库和创建结果集。它相当于把 mysql_select_db 和 mysql_query 两个函数的功能集成到了一起。例如:

```
mysql_select_db("guestbook",$conn);                //选择数据库
$result=mysql_query("Select * from lyb",$conn);    //创建结果集
```

可以用 mysql_db_query()改写为:

```
$result=mysql_db_query('guestbook',"Select * from lyb",$conn);
```

7. 释放结果集 mysql_free_result()

结果集包含的记录会占用服务器内存,虽然在程序代码执行结束后会自动释放结果集占用的内存,但建议在适当时候使用 mysql_free_result()释放内存,例如:

```
mysql_free_result($result);
```

8. 关闭数据库连接 mysql_close()

使用 mysql_close()函数可以关闭使用 mysql_connect()函数建立的连接。例如:

```
mysql_close($conn);
```

10.3.4 使用 mysql_query()增、删、改记录

除了将数据表中的数据显示在页面上,有时还希望通过网页对数据库执行添加、删除或修改操作。比如在网页上发表留言就是向数据表中添加一条记录。

1. 利用 insert 语句添加记录

利用 SQL 语言的 Insert 语句可以执行添加记录操作，而使用 mysql_query 方法实际上可以执行任何 SQL 语句，因此利用该方法执行一条 insert 语句，就可以向数据表中添加一条记录。示例代码如下：

```
----------------------清单 10-3.php 添加记录----------------
<? require('conn.php');
mysql_query("insert into lyb(title, content, author, email,`date`) values('大家好', 'PHP学习园地', '小浣熊', 'sdf@sd.com','2012-3-3')") or die('执行失败');
echo '新增记录的 id是'.mysql_insert_id();          //可选,输出新记录的 id
?>
```

说明：

（1）本程序分为两部分：第一部分是连接数据库，由于连接数据库的代码已写在 conn.php 文件中，因此在这里直接利用 require 函数调用该文件；第二部分是利用 mysql_query 方法添加记录。

（2）mysql_query 只有在执行查询语句时才会返回结果集，在添加记录时不会返回结果集，因此在 mysql_query 前不必写“$result=”，如果写了，则 $result 的值为 false。

（3）用 insert 语句一次只能添加一条记录，如果要添加多条，可以逐条添加或用循环语句。

（4）mysql_insert_id()函数可以返回上一步 insert 查询中新增记录的自动递增字段的值。

（5）die(msg)函数的功能是输出一条消息 msg，并退出当前脚本，等价于 exit()函数。

2. 利用 delete 语句删除记录

当管理员希望删除某些留言时，就需要在数据库中删除记录，可以利用 mysql_query 方法执行一条 delete 语句来删除记录。下面是一个例子。

```
-------------------清单 10-4.php 删除表中的记录----------------
<?    require('conn.php');
mysql_query(" Delete from lyb where ID in(158,162,163,169)") or die('执行失败');
?>
```

本次操作共有<?=mysql_affected_rows() ? >条记录被删除！

mysql_affected_rows()可返回此次操作所影响的记录行数。如果这次有 4 条记录被删除，那么影响的行数就是 4，该函数将返回 4。

提示：使用 mysql_query 方法执行 insert、delete、update 语句，都可以用 mysql_affected_rows()函数返回受影响的记录行数，但如果执行 delete 语句时没有指定 Where 子句（此时所有记录都将被删除），则 mysql_affected_rows()会返回 0，而不是实际被删除的记录数。

3. 利用 update 语句更新记录

当需要修改某条留言时，就需要用 mysql_query 方法执行 update 语句更新记录。例如：

```
----------------------清单 10-5.php 更新表中的记录----------------
<? require('conn.php');
mysql_query("Update lyb set email='rong@163.com', author='蓉蓉' where ID>133 and ID<143") or die('
执行失败');
?>
```

这样将修改符合条件的记录。update 语句常用来记录新闻页面的单击次数，假设单击次数记录在 hits 字段中。只要在显示某条新闻的页面的适当位置加入如下这条语句就可以了。

```
mysql_query("update news set hits=hits+1 where id='".$_GET['id']."'");
```

这样每打开一次这个新闻页面，都会执行这条 SQL 语句，使单击次数（hits 字段）加 1。

10.4 增、删、改记录的综合实例

本节介绍一个综合实例，它能通过网页对数据表中的数据进行添加、删除和修改操作。该程序包括数据记录管理主界面，添加记录模块、删除记录模块和修改记录模块。

10.4.1 管理记录主页面的设计

我们可以对 10-2.php 稍做修改，使其在显示记录的基础上增加添加、删除和修改记录的链接，分别链接到添加、删除和修改记录的 PHP 文件上。将这个网页作为管理留言的首页，命名为 10-6.php，程序代码如下，运行效果如图 10-21 所示。

```
-------------------清单 10-6.php 管理记录的主页面------------------
<? require('conn.php');                          //连接数据库
$result=mysql_query("Select * from lyb",$conn);  //创建结果集
?>
<a href="addform.php">添加记录</a>
<table border="1" width="95%">
  <tr bgcolor="#e0e0e0">
    <th>标题</th><th>内容</th><th>作者</th><th>email</th>
    <th>来自</th><th>删除</th><th>更新</th></tr>
  <? while($row=mysql_fetch_assoc($result)){      //显示结果集中记录
?>
  <tr><td><?=$row['ID'] ?></td><td><?=$row['content'] ?></td>
    <td><?=$row['author'] ?></td><td><?=$row['email'] ?></td>
    <td><?=$row['ip']?></td>
```

```
        <td><a href="delete.php?id=<?=$row['ID'] ?>">删除</a></td>
        <td><a href="editform.php?id=<?=$row['ID'] ?>">更新</a></td>
</tr>
  <? } ?>
</table>
```



图 10-21 程序 10-6.php 的运行效果

说明：

(1) 请注意代码中的“删除”超链接：

```
<a href="delete.php?id=<?=$row['ID'] ?>">删除</a>
```

其中，<?=$row['ID'] ?>会输出这条记录 ID 字段的值，而每条记录的 ID 字段值都不相同，因此，所有记录后的“删除”超链接虽然都是链接到同一页面(delete.php)，但带的 id 参数值不同，这样就可以将这条记录的 id 参数值传递给 delete.php。

例如，如果这条记录的 ID 字段值为 4，则这个超链接实际上为：

```
<a href="delete.php?id=4">删除</a>
```

在 delete.php 中，就可以用 $_GET[]获取这个 id 值。再根据该 id 值，删除对应的记录。对于更新记录的超链接也是同样的道理。

(2) 在有些程序中，删除和更新不是使用的超链接，而是使用表单中的按钮，如果要使用按钮，只要将

```
<a href="editform.php?id=<?=$row['ID'] ?>">更新</a>
```

替换成：

```
<form action="editform.php?id=<?=$row['ID'] ?>" method="post">
      <input type="submit" value="更新">
</form>
```

该表单的作用仅仅是利用 action 属性来传递 URL 参数，表单并没有向处理页提交任何内容。(注意，method 属性不能省略，想一想把 method 属性设置为 get 还可以吗?)

(3) 如果希望用户在单击“删除”链接后弹出一个确认框询问用户是否确定删除，可以将 10-6.php 中“删除”超链接的代码修改为：

```
<a href="delete.php?id=<?=$row['ID'] ?>" onclick="return confirm('确认要删除吗? ')">删除</
a>
```

这样,由于 onclick 事件中的代码会先于 href 属性执行,因此当用户单击超链接时,将先弹出确认框(见图 10-22),如果单击确认框中的"取消"按钮,则 confirm()函数将返回 false,本次单击超链接的行为将失效,就不会再链接到 delete.php 进行删除了。



图 10-22 删除确认框

10.4.2 添加记录的实现

当用户单击图 10-21 中的"添加记录"时,就会转到 addform.php。该网页是个纯静态网页,它含有一个表单,用户可在表单中输入留言内容。其代码如下,运行效果如图 10-23 所示。

```
----------------清单 10-7 addform.php 添加记录的界面-----------------
<h2 align="center">请您在下面填写留言</h2>
<form method="post" action="insert.php">
  <table width="400" border="1" align="center" cellpadding="2">
    <tr><td width="125">留言标题:</td>
      <td width="275"><input type="text" name="title"> * </td></tr>
    <tr><td>留言人:</td>
      <td><input type="text" name="author"> * </td></tr>
    <tr><td>联系方式:</td>
      <td><input type="text" name="email"> * </td></tr>
    <tr><td>留言内容:</td>
      <td><textarea name="content" cols="30" rows="2"></textarea></td></tr>
    <tr><td> </td><td><input type="submit" value="提 交"></td></tr>
</table></form>
```



图 10-23 添加留言 addform.php 的主界面

单击“提交”按钮，表单中的数据就会提交给 insert. php，该程序首先用 $ _POST 获取表单中的数据，然后用 mysql_query 方法执行 insert 语句，将用户输入的数据作为一条记录插入到 lyb 表中。代码如下，执行过程如图 10-24 所示，这样用户的留言就添加到了数据表中。

```
----------------清单 10-8 insert.php 添加记录的主程序----------------
<? //ob_start();
require('conn.php');
$title=$_POST["title"];            //获取表单元素的值
$author=$_POST["author"];
$email=$_POST["email"];
$content=$_POST["content"];
$ip=$_SERVER['REMOTE_ADDR'];       //获得客户端 IP 地址
$sql="insert into lyb(title,author,email,content,ip,`date`) values('$title','$author','$email', '$content
','$ip','date(Y-m-d h:i:s) ')";
//echo $sql;                       输出 sql 语句，用于调试，可删除
mysql_query($sql) or die('执行失败');
header("Location:10-6.php");       //插入成功后，自动转到首页
?>
```

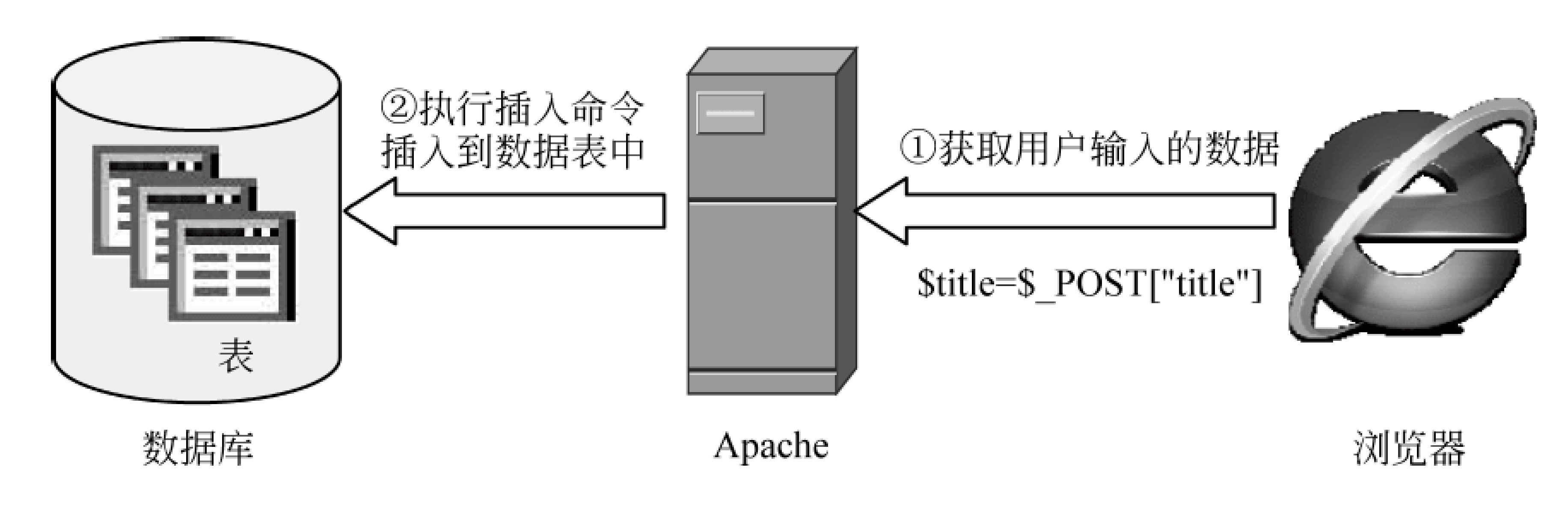


图 10-24　添加记录的步骤

说明：

（1）该程序中的 insert 语句较长，因此将其放在一个变量（$ sql）中。这样做的另一个好处是，如果 SQL 语句有错误，则可以先输出该 SQL 语句，便于调试。

（2）对于记录中自动递增字段的值（如 ID 字段），系统会自动生成，切忌不要用 insert 语句插入自动递增字段的值，否则易引起错误。

10.4.3　删除记录的实现

在图 10-21 中单击“删除”链接时，就会执行 delete. php 程序，该程序先获取从超链接传递过来的记录 id 参数，然后用 delete 语句删除 id 对应的记录，过程如图 10-25 所示。

```
-----------------清单 10-9 delete.php 删除记录的主程序-----------------
<?     require('conn.php');
$id=intval($_GET['id']);               //获取 10-6.php 传来的 id 参数并转换为整型
$sql="delete from lyb where ID=$id";
```

```
if(mysql_query($sql) && mysql_affected_rows()==1)        //执行 sql 语句并判断执行
                                                          //是否成功
    echo "<script>alert('删除成功!');location.href='10-6.php'</script>";
else
    echo "<script>alert('删除失败!');location.href='10-6.php'</script>";
?>
```

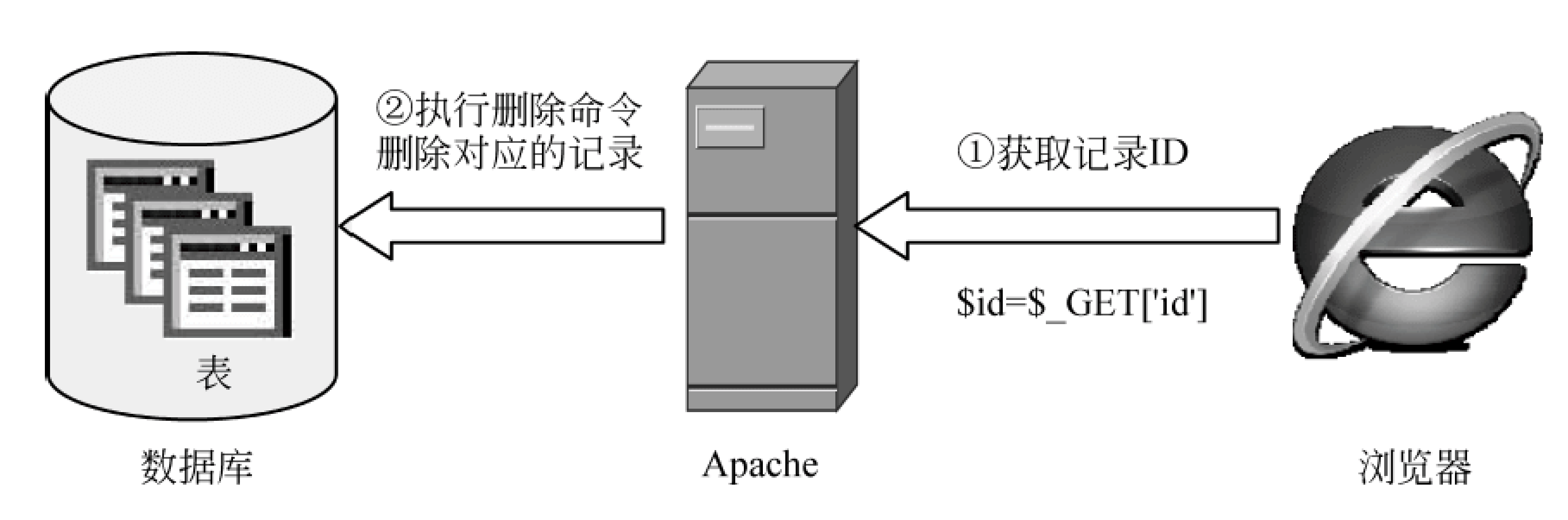


图 10-25　删除记录的步骤

说明：

(1) 在第二行中使用了 intval 函数将获取到的 id 参数强制转换为整型，虽然在一般情况下不转换也可以，但这样做的好处是可以防止非法用户在浏览器地址栏中手工输入一些非数值型的 id 参数，如"id='"破坏系统。

(2) 如果 mysql_query 函数执行了一条合法的 SQL 语句(无论是否有记录被删除)，那么该函数将返回 true，否则返回 false。因此 mysql_query 返回 true 并不表示一定有记录被删除。为此程序采用 mysql_affected_rows()判断是否有记录被删除。

10.4.4　同时删除多条记录的实现

在电子邮件系统中，允许用户选中多封邮件后将其一并删除，这就是同时删除多条记录的例子。我们可以对图 10-21 中的 10-6.php 做些修改，将每条记录后的"删除"超链接换成一个多选框，再在最后一行添加一个"删除"按钮，代码如下，运行结果如图 10-26 所示。

```
-----------清单 10-10 delall.php 同时删除多条记录的程序-----------------
<?  require('conn.php');
if($_GET["del"]==1){                          //如果用户按了"删除"按钮
        $selectid=$_POST["selected"];         //获取所有选中多选框的值,保存到数组中
    if(count($selectid)>0){                   //防止 selectid 值为空时执行 SQL 语句出错
        $sel=implode(',',$selectid);          //将各个数组元素用","号连接起来
        mysql_query("delete From lyb where ID in($sel)") or die('执行失败');
        header("Location:delall.php");        //删除完毕,刷新页面
        }
    else echo '没有被选中的记录';
}
$result=mysql_query("Select * from lyb",$conn);           //创建结果集
    ?>
```

```
<form method="post" action="?del=1"><!--表单提交给自身-->
<table border="1" width="95%">
  <tr bgcolor="#e0e0e0">
    <th>标题</th><th>内容……<th>删除</th><th>更新</th></tr>
<? while($row=mysql_fetch_assoc($result)){
?>
  <tr><td><?=$row['title']?></td><td><?=$row['content']?></td>
    <td><?=$row['author']?></td><td><?=$row['email']?></td>
    <td><?=$row['ip']?></td>
    <td align="center">
<input type="checkbox" name="selected[]" value="<?=$row['ID']?>"></td>
<!--复选框-->
    <td><a href="editform.php?id=<?=$row['ID']?>">更新</a></td></tr>
  <? } ?>
<tr bgcolor="#E0E0E0">
    <td></td><td></td><td></td><td></td><td></td>
    <td align="center"><input type="submit" value="删 除"></td><!--删除按钮-->
        <td></td></tr>
</table></form>
```



图 10-26　delall.php 的运行结果

说明：

（1）每条记录后的多选框的 name 属性值是静态的“selected[]”，因此循环以后所有记录多选框的 name 属性值都是 selected[]，而多选框的 value 属性值是动态数据<?=$row['ID']?>，则循环后每条记录多选框的 value 属性值都是其 id 字段值。我们知道，如果有多个多选框的 name 属性值相同，那么提交的数据就是 selected[]=2&selected[]=3& selected[]=5 的形式。因此，在该程序中，如果用户选中多条记录（比如选中 2、3、5 条记录），则 $_POST["selected"]是一个数组：Array([0]=>2[1]=>3[2]=>5)，用 implode 函数将该数组中的元素用逗号连接起来，就得到字符串 $sel=2,3,5。那么最终执行的 SQL 语句就是 Delete From lyb where id in("2,3,5")。这是一条正确的 SQL 语句，因此会删除第 2、3、5 条记录。

（2）本程序将表单界面和删除记录的程序写在了同一个文件中，方法是通过 action

属性将表单提交给自身而不是其他文件，但增加了一个 URL 字符串，处理程序据此判断是否提交了表单。

10.4.5 修改记录的实现

修改记录的过程分为两阶段：

(1) 第一阶段是提供一个显示待修改记录的表单，该表单显示待修改记录各个字段的值，以供用户修改记录中的信息。显示待修改记录的程序 editform.php 代码如下，其程序流程如图 10-27 所示。运行结果如图 10-28 所示。

```
----------清单 10-11 editform.php 显示待修改记录的程序-----------------
<?  require('conn.php');
$id=intval($_GET['id']);                    //将获取的 id强制转换为整型
$sql="Select * from lyb where ID=$id";      //获取待更新的记录
$result=mysql_query($sql,$conn);
$row=mysql_fetch_assoc($result);            //将待更新记录各字段的值存入数组中
?>
<h2 align="center">更新留言</h2>
<form method="post" action="edit.php?id=<?=$row['ID'] ?>">
  <table width="400" border="1" align="center" cellpadding="2">
    <tr><td width="125">留言标题:</td>
      <td width="275"><input type="text" name="title" value="<?=$row['title'] ?>"> * </td>
    </tr>
    <tr><td>留言人:</td>
      <td><input type="text" name="author" value="<?=$row['author'] ?>"> * </td></tr>
    <tr><td>联系方式:</td>
      <td><input type="text" name="email" value="<?=$row['email'] ?>"> *
      </td></tr>
    <tr><td>留言内容:</td>
      <td><textarea name="content" cols="30" rows="2"><?=$row['content']?>
      </textarea>
</td></tr>
    <tr><td> </td><td><input type="submit" value="确 定"></td></tr>
  </table></form>
```


图 10-27 修改记录的过程(第一阶段)



图 10-28　程序 editform.php 的运行结果

说明：

① 该程序界面和 addform.php 的界面很相似，但区别是表单中显示了一条记录的信息。它首先根据首页传过来的 id 值，执行查询找到这条记录，然后将其显示在表单中，由于只有一条记录，所以不需要用到循环语句。

② 请注意将动态数据显示在表单中的方法。对于单行文本框，它在初始时会显示 value 属性中的值，因此只要给其 value 属性赋值就可以了，如 value="<?= $row['title'] ?>"；对于多行文本域，它在初始时会显示标记中的内容，因此将动态数据写在标记中即可。

③ 表单传递 id 给表单处理程序有两种方法：一是使用上述代码中的 URL 字符串方式（action="edit.php?id=<?= $row['ID'];?>"）；二是使用表单隐藏域传递，比如在 editform.php 的表单中添加一个隐藏域：

```
<input type="hidden" name="id" value="<?=$row['ID'];?>" />
```

（2）修改记录的第二阶段是：当用户单击“确定”按钮提交表单后，浏览器将与服务器进行第二次通信。修改记录处理程序 edit.php 首先获取从<form>标记中传递过来的 id 值，并获取表单中填写的数据，根据 id 值和表单数据修改该 id 对应的记录，其过程如图 10-29 所示。

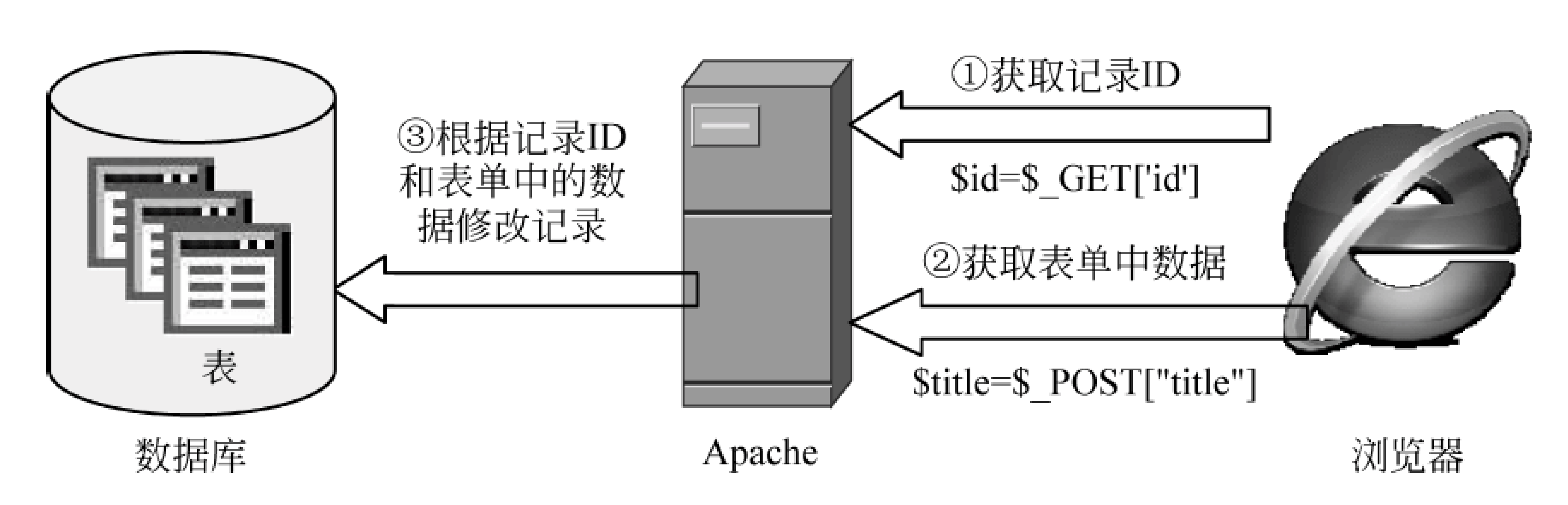


图 10-29　修改记录的过程（第二阶段）

修改记录的执行程序（edit.php）的代码如下：

---------------清单 10-12 edit.php 修改记录的执行程序-----------------

```
<? require('conn.php');
$id=intval($_GET['id']);                    //获取记录 id
$title=$_POST["title"];                     //获取表单中数据
$author=$_POST["author"];
$email=$_POST["email"];
$content=$_POST["content"];
$ip=$_SERVER['REMOTE_ADDR'];                //获取客户端 IP
$sql="Update lyb Set title='$title',author='$author',email='$email',content=
'$content' Where ID=$id";
mysql_query($sql) or die('执行失败');
echo "<script>alert('留言修改成功!');location.href='10-6.php';</script>";
?>
```

说明：

① Update 语句根据传过来的 id 找到要修改的留言；

② 更新完成后本程序采用输出客户端脚本的方法(location.href)转向首页，用来替代 header 语句，这样做的好处是可以在返回之前弹出一个警告框提示用户“留言修改成功”，而 header 方法则无法在转向之前输出任何警告框之类的 JavaScript 的脚本，想一想为什么。因此前面几个程序的 header 语句都可以换成这句，以增加弹出警告框提示用户的功能。

10.4.6 查询记录的实现

动态网站的一个明显优势是能够让用户在网站内快速搜索到符合条件的内容，如搜索某种商品、某条新闻等。如果网站中所有的记录都存放在同一个表中，则只要借助于 Select 语句的模糊查询功能，就能方便地按照条件查询到相关记录。

查询记录程序的流程如下：首先提供一个文本框供用户输入要查询的关键字，然后将用户提交的关键字作为条件用 Select 语句进行查询，最后将查询的结果(返回的结果集)显示在网页中。下面在程序 10-2.php 的基础上添加查询功能，首先在该文件的<table>标记前加入如下表单代码，修改后的页面(10-7.php)如图 10-30 所示。

```
<form method="get" action="10-8.php">
<div style="border:1px solid gray; background:#eee;padding:4px;">
查找留言：请输入关键字<input name="keyword" type="text">
  <select name="sel">
    <option value="title">文章标题</option>
    <option value="content">文章内容</option>
  </select>
  <input type="submit" value="查询">
</div></form>
```

处理查询的程序 10-8.php 的代码如下：

```
<h3 align="center">查询结果</h3>
```



图 10-30　查询记录的界面

```
<?  require('conn.php');
$keyword=trim($_GET['keyword']);          //获取输入的关键字
$sel=$_GET['sel'];                        //获取选择的查询方式
$sql="select * from lyb";
if($keyword<>"")
    $sql=$sql ." where $sel like '%$keyword%'";     //构造查询语句
$rs=mysql_query($sql) or die('执行失败');
if(mysql_num_rows($rs)>0) {
    echo "<p>关键字为"$keyword",共找到".mysql_num_rows($rs)." 条留言</p>"; ?>
<table border="1">
  <tr bgcolor="#e0e0e0">
    <th>标题</th><th width="100">内容</th><th width="60">作者</th>
    <th>email</th><th width="80">来自</th></tr>
  <? while($row=mysql_fetch_assoc($rs)){
?>
  <tr><td><?=$row['title'] ?></td><td><?=$row['content'] ?></td>
    <td><?=$row['author'] ?></td><td><?=$row['email'] ?></td>
    <td><?=$row['ip'] ?></td></tr>
  <? }}
else   echo "没有搜索到任何留言";    ?>
</table>
```

在图 10-30 的查询框中输入“大家”，则该程序的运行结果如图 10-31 所示。



图 10-31　10-8.php 的运行结果

说明：该程序可根据“标题”或“内容”进行查询，只要在下拉列表框中进行选择，就会将文本框中的内容发送给服务器，作为 title 字段或 content 字段，以构造相应的查询语句。

10.5 分页显示数据

分页是一种将所有信息分段显示给浏览器用户的技术。用户每次看到的不是全部信息,而是其中一部分信息,如果用户没有找到自己想要的内容,就可以使用翻页链接切换可见内容。分页显示功能在新闻列表页、论坛或留言板等程序中广泛存在。当记录很多时,程序能自动将结果集分页显示,用户可以一页一页地浏览。例如在图 10-32 中,结果集中共有 14 条记录,每页显示 4 条,这样就分成了 4 页。

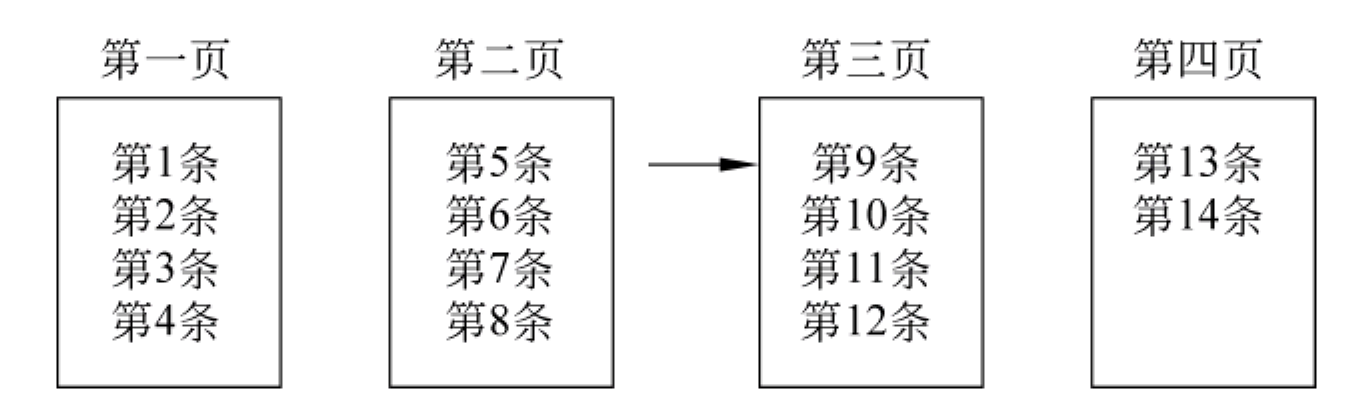


图 10-32 分页显示记录示意图

在 B/S 程序中,分页技术可以分别在数据库服务器、Web 服务器或浏览器中实现。如果通过创建结果集的方式来分页,就是在数据库服务器上实现分页的;如果通过 PHP 循环语句读取结果集中某页范围的记录,就是在 Web 服务器上实现分页的;如果通过客户端 JavaScript 脚本只显示某页记录对应的 HTML 元素,就是在浏览器中实现的分页。

10.5.1 分页程序的基本实现

分页程序实现的步骤大致是:

①设置每页显示的记录数→②获取记录总数→③计算总共有多少页→④取得要显示第几页的记录→⑤通过超链接传递页码。

1. 设置每页显示的记录数量

假定使用变量 $PageSize 来保存每页显示的记录数,它的值由用户根据需要自行设置。例如,设置每页显示 4 条记录,可以使用下面的语句:

```
$PageSize=4;
```

2. 获取结果集中的记录总数

获取结果集中记录总数有两种方法。第一种是通过 mysql_num_rows()函数返回记录总数,并将其保存在 $RecordCount 变量中。代码如下:

```
$RecordCount=mysql_num_rows($result);
```

第二种方法是通过 select 语句中的 count 函数实现。代码如下:

```
$result=mysql_query("Select count(*) from lyb",$conn);            //统计记录数量的结果集
$row=mysql_fetch_row($result);
```

```
$RecordCount=$row[0];
```

3. 计算总页数

可以通过$RecordCount和$PageSize两个变量的值计算得到总页面数$PageCount，方法如下：

```
$PageCount=ceil($RecordCount/$PageSize);
```

说明：ceil(x)函数用来返回大于或等于x并且最接近x的整数。如果结果集中的记录总数$RecordCount是$PageSize的整数倍，则$PageCount等于$RecordCount除以$PageSize的结果；否则，$PageCount等于$RecordCount除以$ PageSize的结果取整后加1。

4. 如何显示第n页的记录

虽然使用$PageSize可以控制每页显示的记录数，但是要显示哪页的记录呢？这可以在Select语句中使用limit子句限定显示记录的范围，方法如下：

```
Select * from 表名 limit 起始位置,显示记录数量
```

例如，用$Page保存当前页码，要获取第$Page页显示的记录，SQL语句如下：

```
Select * from 表名 limit($Page-1) * $PageSize,$PageSize
```

这样，只要给$Page赋一个值n，就能显示第n页的记录了。代码如下：

```
<? require('conn.php');
$Page=3;                    //显示第3页的记录
$PageSize=4;
$result=mysql_query("Select * from lyb",$conn);    //创建获取记录总数的结果集
$RecordCount=mysql_num_rows($result);
$PageCount=ceil($RecordCount/$PageSize);
$result=mysql_query("Select * from lyb limit ".($Page-1) * $PageSize.", ".$PageSize,$conn);   ?>
<table border="1" width="95%"><tr bgcolor="#e0e0e0">
<th>标题</th><th>内容</th><th>作者</th><th>email</th><th>来自</th></tr>
  <? while($row=mysql_fetch_assoc($result)){   ?>
  <tr><td><?=$row['ID'] ?></td><td><?=$row['content'] ?></td>
    <td><?=$row['author'] ?></td><td><?=$row['email'] ?></td>
    <td><?=$row['ip'] ?></td></tr>
  <? }
  mysql_free_result($result);   ?></table>
```

说明：limit子句中的记录序号从0开始，第1条记录的序号为0。因此($Page－1)*$PageSize就表示前n－1页的所有记录再加1，正好是第n页的第1条记录。

5. 如何通过超链接转到要显示的分页

可以通过超链接传递参数的方法通知脚本程序要显示的页码。假定分页显示记录的

页面是 10-9. php,则传递参数的链接如下:

```
http://localhost/php/10-9.php?page=2
```

参数 page 用来指定当前的页码。在 10-9. php 中,可以通过下面的语句读取参数 page:

```
if(isset($_GET['page']))          //如果获取到的页码不为空
       $Page=$_GET['page'];
else     $Page=1;
```

这样 $ Page 就保存了 URL 中的页码。但普通用户不会知道在 URL 上输入类似 ?page=2 之类的参数来访问分页。为此,我们可以定义几个分页链接,供用户单击。

"首页""上一页""下一页""末页"的链接的代码分别为:

```
echo "<a href='?page=1'>首页</a>";                //转到当前页的第一页
echo "<a href='?page=" . ($Page-1) . "'>上一页</a>";
echo "<a href='?page=" . ($Page+1) . "'>下一页</a>";
echo  "<a href='?page=" .$PageCount . "'>末页</a>";
```

在比较完美的分页程序中,还需要根据当前页对翻页链接进行判断控制。如果当前页码是 1,则取消"第一页"和"上一页"的链接;如果当前页是最后一页,则取消"下一页"和"末页"的链接。

下面是一个分页显示程序的完整代码,它的运行效果如图 10-33 所示。

```
---------------------清单 10-13.php 分页显示记录程序 1-----------------
<?     require('conn.php');
if(isset($_GET['page']) && (int)$_GET['page']>0)               //获取页码并检查是否非法
         $Page=$_GET['page'];
else     $Page=1;              //如果获取不到页码则显示第 1 页
//设置每页显示记录数
$PageSize=4;
$result=mysql_query("Select count(ID) from lyb",$conn);          //创建统计记录总数的结果集
$row=mysql_fetch_row($result);
$RecordCount=$row[0];      //获取记录总数
    //计算总共有多少页
$PageCount=ceil($RecordCount/$PageSize);
         //将某一页的记录放入结果集
$result=mysql_query("Select * from lyb limit ". ($Page-1) * $PageSize.", ".$PageSize,$conn);
    //显示记录     ?>
<h3 align="center">分页显示记录</h3>
<table border="1" width="95%">
  <tr bgcolor="#e0e0e0">
    <th>标题</th><th>内容</th><th>作者</th><th>email</th><th>来自</th></tr>
  <? while($row=mysql_fetch_assoc($result)){   ?>
  <tr><td><?=$row['title'] ?></td><td><?=$row['content'] ?></td>
```

```
    <td><?=$row['author'] ?></td><td><?=$row['email'] ?></td>
    <td><?=$row['ip'] ?></td></tr>
  <? }
  mysql_free_result($result);  ?>
</table>
<p><?                                          //显示分页链接的代码
if($Page==1)                                   //如果是第 1 页,则不显示第 1 页的链接
        echo  "第一页  上一页 ";
else echo "<a href='?page=1'>第一页</a><a href='?page=".($Page-1)."'>上一页</a>";
for($i=1;$i<=$PageCount;$i++)    {             //设置数字页码的链接
        if($i==$Page) echo "$i  ";             //如果是某页,则不显示某页的链接
        else echo "<a href='?page=$i'>$i</a>";}
if($Page==$PageCount)                          //设置"下一页"链接
        echo  "下一页  末页 ";
else echo "<a href='?page=" . ($Page+1) . "'>下一页</a>
<a href='?page=" .$PageCount . "'>末页</a>";
echo "   共".$RecordCount. "条记录  ";     //共多少条记录
echo "$Page /$PageCount 页";                    //当前页的位置
?></p>
```



图 10-33　分页显示记录示例

6. 通过移动结果集指针进行分页(在 Web 服务器实现分页)

分页程序的写法其实还有别的办法。例如,要实现只显示第 n 页的记录,除了使用 limit 子句创建仅含有一页记录的结果集外,还可以创建包含所有记录的结果集,并将结果集的指针指向第 n 页的第 1 条记录,然后用 for 循环循环输出 $PageSize 条记录。代码如下:

```
--------------------清单 10-14.php 分页显示记录程序 2-----------------
<? require('conn.php');
if(isset($_GET['page']) && (int)$_GET['page']>0)      //获取页码
        $Page=$_GET['page'];
else    $Page=1;
//设置每页显示记录数
```

```
$PageSize=4;
$result=mysql_query("Select ID from lyb",$conn);     //创建结果集
$RecordCount=mysql_num_rows($result);                //获取记录总数
mysql_free_result($result);
    //计算有多少页
$PageCount=ceil($RecordCount/$PageSize);
        //将所有记录放入结果集
$result=mysql_query("Select * from lyb",$conn);
    //显示记录          ?>
<table border="1" width="95%"><tr bgcolor="#e0e0e0">
    <th>标题</th><th>内容</th><th>作者</th><th>email</th><th>来自</th></tr>
  <?           //将指针指向第$Page页第1条记录
  mysql_data_seek($result,($Page-1)*$PageSize);
  for($i=0;$i<$PageSize;$i++){
        $row=mysql_fetch_assoc($result);
      if($row){           //如果记录不为空,用来处理末页的情况
?>
  <tr><td><?=$row['ID'] ?></td><td><?=$row['content'] ?></td>
    <td><?=$row['author'] ?></td><td><?=$row['email'] ?></td>
    <td><?=$row['ip'] ?></td></tr>
  <?}}
  mysql_free_result($result);  ?>
</table>
<p><?   …        //此处为显示分页链接的代码,与10-13.php相同,故省略
?></p>
```

其中,mysql_data_seek(result, row)函数的功能是将结果集 result 的指针移动到指定的行数 row(行数从 0 开始)。

这种方法创建的结果集中包含了所有记录,也就是说,包含了所有页的记录。从效率上说,如果结果集中记录很多的话,该方法创建的结果集将占用很多的服务器内存,因此效率较低。但如果结果集中记录比较少,则这种方法不必每显示一个分页就执行一个不同的查询,因此效率反而高些。

10.5.2 对查询结果进行分页

10.3.1 节中给出的只是最基本的分页程序。假设要对图 10-31 中搜索留言得到的结果进行分页,则上述分页程序只能正确显示第 1 页,当用户转到其他页后又会显示所有的记录,而不是查询得到的记录。这是因为单击分页链接后没有将用户输入的查询关键字传递给其他页。

为此,可以在获取了用户输入的关键字后,一方面将它传递给 SQL 语句进行查询;另一方面将其保存在分页链接的 URL 参数(或表单隐藏域)中。具体来说,可以给分页链接增加一个 URL 参数,将该 URL 参数的值设置为查询关键字以传递给其他页。关键代码如下:

```
<%  $keyword=trim($_GET['keyword']);                    //获取查询关键字
if($keyword<>"")
        $sql=$sql ." where title like '%$keyword%'";
…
for($i=1;$i<=$PageCount;$i++)     {                    //设置每页的链接
        if($i==$Page) echo "$i  ";
        else              //在此处添加 URL参数 keyword,用来保存关键字
        echo "<a href='?page=$i&keyword=$keyword'>$i</a>";}     ?>
```

这样,单击分页链接时,都会将关键字重新传给 SQL 语句,因此单击分页链接后转到的新分页仍然是查询结果。它的完整代码如下,运行结果如图 10-34 所示。

```
<?    require('conn.php');
if(isset($_GET['page']) && (int)$_GET['page']>0)        //获取页码
        $Page=$_GET['page'];
else    $Page=1;
$PageSize=4;                                        //设置每页显示记录数
$keyword=trim($_GET['keyword']);                    //获取查询关键字
$sql="select * from lyb";
if($keyword<>"")
        $sql=$sql ." where title like '%$keyword%'";
$result=mysql_query($sql,$conn);                    //根据有无查询关键字创建结果集
$RecordCount=mysql_num_rows($result);               //获得记录总数
$PageCount=ceil($RecordCount/$PageSize);            //获得总页数
?>
<form method="get" action="">
<div style="border:1px solid gray; background:#eee;padding:4px;">
查找留言:请输入关键字<input name="keyword" type="text" value="<?=$keyword?>">
  <input type="submit" value="查询">
</div></form>
<table border="1" width="95%">
  <tr bgcolor="#e0e0e0"><th>标题</th>…(省略显示表头的代码)</tr>
<?
mysql_data_seek($result,($Page-1)*$PageSize);              //将指针指向第$Page页第1条记录
for($i=0;$i<$PageSize;$i++){
        $row=mysql_fetch_assoc($result);
     if($row){      ?>
          <tr><td><?=$row['title'] ?></td><td><?=$row['content'] ?></td>
          <td><?=$row['author'] ?></td><td><?=$row['email'] ?></td>
            <td><?=$row['ip'] ?></td></tr>
  <? }} ?>
</table>
<p><?          //显示分页链接
     if($Page==1)    echo  "第一页 上一页 ";
else   echo   "<a href='?page=1&keyword=$keyword'>第一页</a>
```

```
<a href='?page=".($Page-1)."&keyword=$keyword'>上一页</a>";
for($i=1;$i<=$PageCount;$i++)     {                //设置数字页码的链接
        if($i==$Page) echo "$i  ";
        else echo "<a href='?page=$i&keyword=$keyword'>$i</a>";}
if($Page==$PageCount)   echo  "下一页  末页 ";
else echo  "<a href='?page=".($Page+1)."&keyword=$keyword'>下一页</a>
<a href='?page=".$PageCount."&keyword=$keyword'>末页</a>";
echo "&nbsp共".$RecordCount."条记录  ";          //共多少条记录
echo "$Page/$PageCount 页";                        //当前页的位置
?></p>
```

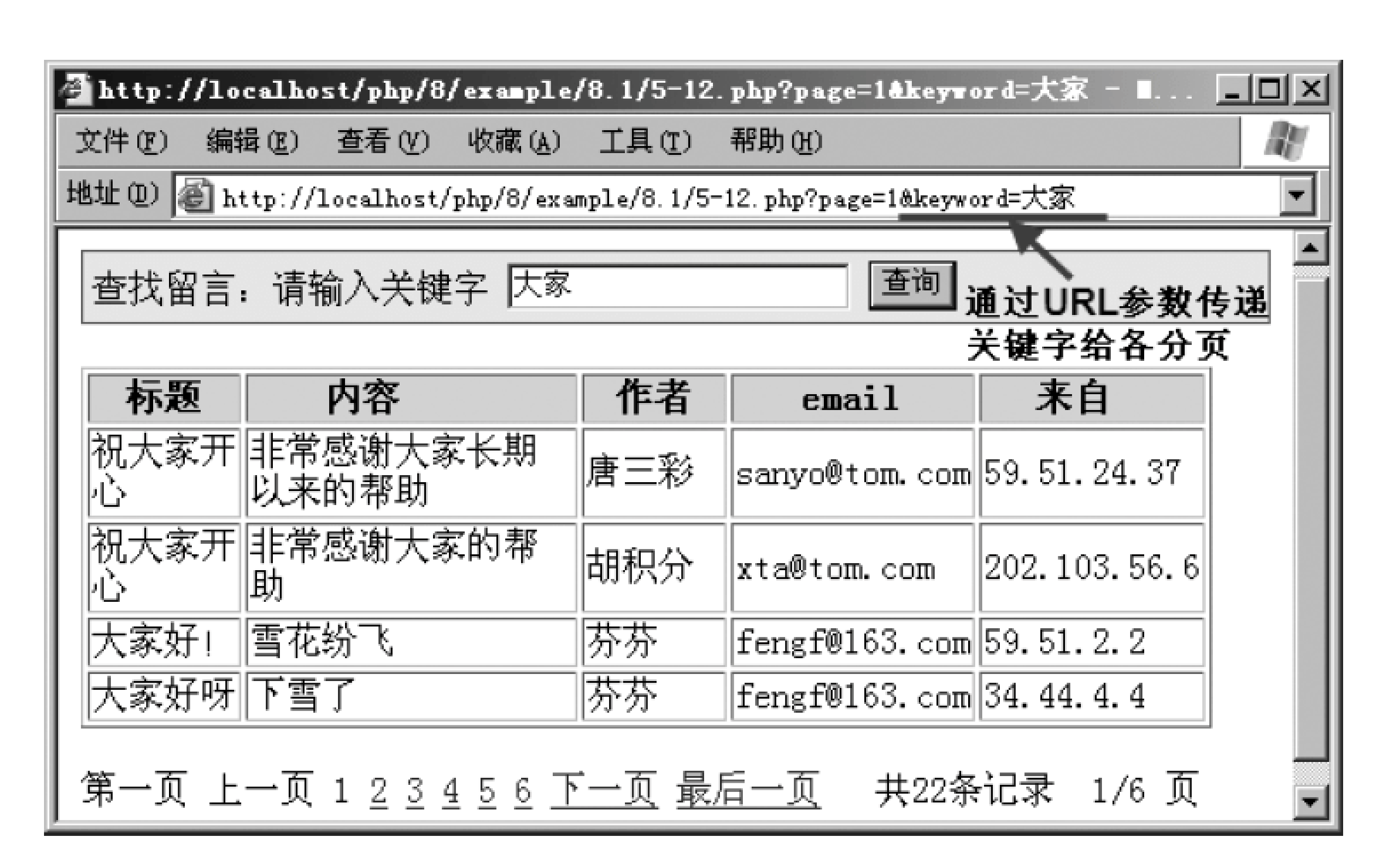


图 10-34 对查询结果进行分页的效果

10.5.3 可设置每页显示记录数的分页程序

在有些分页程序中，还具有让用户选择每页显示多少条记录的功能，如图 10-35 所示。对于记录数非常多的网页来说，这种功能对用户来说更友好。

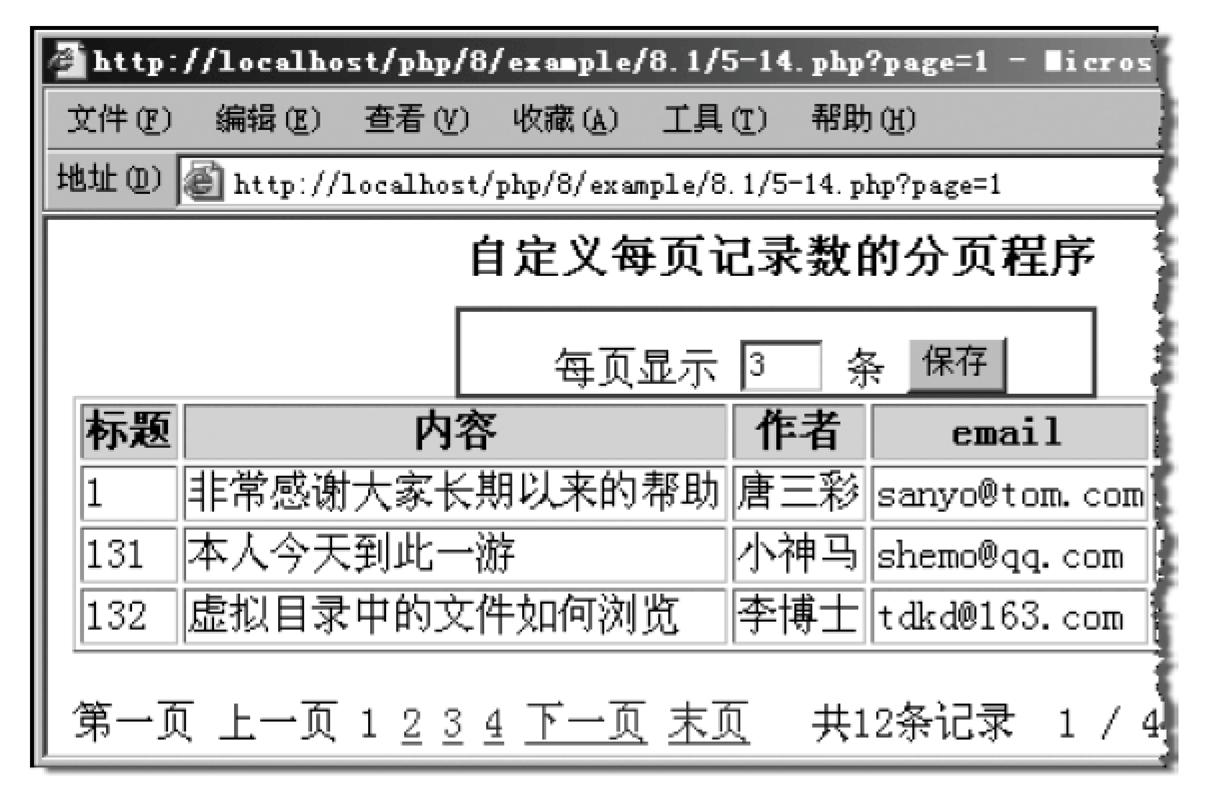


图 10-35 可设置每页显示多少条记录的分页程序

要实现该功能，首先在网页中添加一个表单，表单的文本框中可输入每页显示的记录

数。如果用户提交了表单，就把用户设置的每页记录数赋给 $ PageSize 变量，这样分页程序就会根据新的 $ PageSize 值重新分页。但转到其他页后，由于获取不到用户设置的记录数，$ PageSize 的值又会恢复成默认值。为此，应该把用户设置的记录数保存起来，可以采用 10.5.2 节介绍的方法将该值保存到 URL 参数中，也可以将该值保存到一个 session 变量中，这样其他分页都能获取到用户设置的记录数。本例采用第二种方法，代码如下，运行效果如图 10-35 所示。

```
--------------------清单 10-16.php----------------
<? session_start();
require('conn.php');
if(isset($_GET['page']) && (int)$_GET['page']>0)        //获取页码
        $Page=$_GET['page'];
else    $Page=1;
//设置每页显示记录数,并将记录数保存到 Session 变量中
if(isset($_GET['pagesize'])){                          //如果用户设置了每页记录数
        $PageSize=$_GET['pagesize'];                   //将用户设置的值赋给$PageSize
        $_SESSION["pagezize"]=$_GET['pagesize'];       //将该值保存到 Session 变量中
}
if($_SESSION["pagezize"]<>"")                          //如果 SESSION 值不为空
        $PageSize=$_SESSION["pagezize"];
else
        $PageSize=4;            //第一次打开网页时默认每页显示 4 条
$result=mysql_query("Select * from lyb",$conn);       //创建结果集
$RecordCount=mysql_num_rows($result);
$PageCount=ceil($RecordCount/$PageSize);               //计算有多少页
    //显示记录      ?>
<h3 align="center">自定义每页记录数的分页程序</h3>
<form style="margin:0 auto; text-align:center;" method="get" action="">每页显示<input type="
text" name="pagesize" size="3" value="<?=$PageSize?>">条<input type="submit"  value="保存"><
/form>
<table border="1" width="95%">
  <tr bgcolor="#e0e0e0">
    <th>标题</th><th>内容</th><th>作者</th><th>email</th><th>来自</th>
    </tr>
  <?
  mysql_data_seek($result,($Page-1)*$PageSize);                 //将指针指到某页的第 1条记录
  for($i=0;$i<$PageSize;$i++){
        $row=mysql_fetch_assoc($result);
            if($row){       ?>
  <tr><td><?=$row['ID'];?></td><td><?=$row['content'];?></td>
    <td><?=$row['author'];?></td><td><?=$row['email'];?></td>
    <td><?=$row['ip'];?></td>
</tr>
  <? } }
```

```
  mysql_free_result($result);  ?>
</table>
<p><? …        //此处为显示分页链接的代码,与 10-13.php 相同,故省略
?></p>
```

10.6 mysqli 扩展函数的使用

从 PHP 5.0 开始,不仅可以使用原来的 mysql 函数,而且还可以使用新的 mysqli 扩展函数实现与 MySQL 数据库的信息交流。mysqli 被封装到一个类中,它是一种面向对象的技术,其中 i 表示改进(improvement),其执行速度更快。大多数 mysqli 函数的函数名与 mysql 函数名类似,只不过将函数名的前缀由"mysql"改成了"mysqli"。

要在 PHP 中使用 mysqli 函数,需要在 php.ini 中进行配置。找到下面的配置项:

```
;extension=php_mysqli.dll
```

去掉前面的注释符(;),保存更改,重启 Apache 服务,就可以使用 mysqli 函数了。

10.6.1 连接 MySQL 数据库

在 mysqli 中,提供了两种方法创建到数据库的连接。

1. 使用 mysqli_connect()函数

mysqli_connect()函数用来连接 MySQL 数据库,语法如下:

```
mysqli 对象名=mysqli_connect(数据库服务器, 用户名, 密码, 数据库名)
```

例如,要访问本机上的 guestbook 数据库,用户名为 root,密码为 111。代码如下:

```
$conn=mysqli_connect('localhost','root','111','guestbook');
```

可见,mysqli_connect 函数比 mysql_connect 多了一个参数,使它在连接数据库服务器的同时还能选择数据库。

2. 声明 mysqli 对象

可以使用声明 mysqli 对象的方法来创建连接对象,方法如下:

```
$conn=new mysqli('localhost','root','111','guestbook');
```

如果在创建 mysqli 对象时没有向构造方法传入参数,则需要多写几行代码,包括调用 connect 函数连接数据库服务器,使用 select_db 方法选择数据库。代码如下:

```
$conn=new mysqli();
$conn->connect('localhost','root','111');
$conn->select_db('guestbook');
```

说明:

（1）new mysqli()表示新建一个 mysqli 类的实例，类的实例即对象，赋给变量 $ conn。

（2）由于 $ conn 是一个对象，调用对象的方法可以使用“对象名->方法名”的形式。

（3）mysqli 扩展模块包括 mysqli、mysqli_result 和 mysqli_stmt 三个类。$ conn 就是一个 mysqli 对象，它属于 mysqli 类。mysqli 类常见的成员方法如表 10-1 所示。

表 10-1　mysqli 类中的成员方法

方　法　名	功　　能
connect()	打开一个新的连接到 MySQL 数据库服务器
select_db()	选择当前数据库
set_charset()	设置客户端的默认字符集
close()	关闭先前打开的连接
query()	执行 SQL 语句，并返回结果集(对于 Select 语句)或不返回
multi_query()	同时执行多个查询语句
store_result()	在执行多查询语句时，获取当前结果集
next_result()	在执行多查询语句时，获取当前结果集的下一个结果集
more_results()	从多查询语句中检查是否有任何更多的查询结果集

使用这些成员方法可以对数据库进行查询、创建结果集等操作。

10.6.2　执行 SQL 语句创建结果集

mysqli_query()函数或 mysqli 对象的 query()函数都可用来执行 SQL 语句，如果执行的是 Select 语句，则会返回一个结果集；如果执行的是 insert、delete 等非查询语句，则不会返回结果集。

（1）mysqli_query()的语法如下：

```
结果集=mysqli_query(连接对象, SQL语句)
```

可见，mysqli_query()两个参数的顺序与 mysql_query()的参数顺序相反。例如：

```
$result=mysqli_query($conn,'select * from lyb');
```

（2）mysqli 对象的 query()函数的基本语法和示例如下：

```
对象名->query(SQL语句)
    $result=$conn->query('select * from lyb');
```

结果集中的所有数据默认是都发送到客户端的，如果希望把结果集暂存在 MySQL 服务器上，在有需要时才一条条地读取记录过来，就需要在调用 query 方法时，使用它的第二个参数，并提供一个 MYSQLI_USE_RESULT 值。例如：

```
$result=$conn->query('select * from lyb', MYSQLI_USE_RESULT);
```

在结果集比较大或不适合一次全部取回到客户端时,使用这个参数比较有用。

(3) 执行非查询语句。

如果执行的是非查询语句,则不会返回结果集,因此不要将函数的返回值赋给 $ result。例如:

```
$conn->query('delete from lyb where ID=3');
```

10.6.3 从结果集中获取数据

结果集实际上是 mysqli_result 类的一个对象,可以使用 mysqli_result 类中的一些方法获取结果集中的数据。如果要获取结果集中的当前记录并存储到数组,可使用如下代码:

```
$row=$result->fetch_assoc();
```

这样就将指针指向的当前记录保存到数组 $ row 中,并使结果集指针指向下一条记录。

下面的例子从结果集中获取所有记录,并输出到表格中,运行效果如图 10-20 所示。

```
---------------------清单 10-17.php-----------------
<?  $conn=new mysqli();
    $conn->connect('localhost','root','111');
    $conn->select_db('guestbook');             //连接数据库
    $conn->query('set names gb2312');          //设置字符集
    $result=$conn->query('select * from lyb'); //创建结果集
?>
<table border="1" width="95%">
    <tr bgcolor="#e0e0e0">
    <th>标题</th><th width="100">内容</th><th width="60">作者</th>
    <th>email</th></tr>
  <? $result->data_seek(5);                    //从第 6条记录开始读,可去掉
while($row=$result->fetch_assoc()){            //循环读取结果集中的记录
?>
  <tr><td><?=$row['title'] ?></td><td><?=$row['content'] ?></td>
    <td><?=$row['author'] ?></td><td><?=$row['email'] ?></td></tr>
  <? } ?>
</table>
<p>记录总数<?=$result->num_rows ?></p>
```

上例中 num_rows 是 mysqli_result 类中的一个成员属性,用来返回结果集中的记录总数。而 fetch_assoc()是 mysqli_result 类中的一个成员方法,mysqli_result 类中常用方法如表 10-2 所示。

表 10-2 mysqli_result 类中的成员方法

方 法 名	功 能

fetch_row()	以索引数组的形式返回结果集中当前指向的记录
fetch_assoc()	以关联数组的形式返回结果集中当前指向的记录
fetch_array()	以索引数组和关联数组的形式返回结果集中当前指向的记录
fetch_object()	以对象的形式返回结果集中当前指向的记录
data_seek(n)	将结果集指针指向第 n 条记录
fetch_field()	从结果集中获得某一字段的信息
fetch_fields()	从结果集中获得全部字段的信息
field_seek()	设置结果集中字段的偏移位置
close()	关闭结果集

提示：如果要判断结果集不为空，只能使用 if($result->num_rows>0)来判断，而不能使用 if($result)来判断，因为 $conn->query()只有在执行查询出错时才会返回 false；如果执行查询正确，即使查询结果只有 0 条记录，也会返回一个成员为 0 的对象。

10.7　用 mysqli 制作新闻网站

本节将把如图 10-36 所示的一个静态网页转化为动态网站，也就是向静态网页中绑定数据。由于制作一个完整的动态网站要经过数据库设计、制作前台页面、制作后台管理程序等步骤，工作量相当大。因此在实际中，我们一般是借用别人的数据库和后台管理程序，用于添加、删除和修改网站中的新闻内容，这样的后台管理系统称为 CMS(内容管理系统)或新闻管理系统。自己只制作前台页面(主要包括首页、栏目首页和内页三个页面)，然后在这些页面中绑定动态数据(即显示数据库中的有关数据)。

10.7.1　为网站引用后台程序和数据库

这里以风诺新闻系统为例，介绍在制作网站时如何利用它的数据库和后台程序。首先在百度上搜索“风诺新闻系统”，下载下来后将其所有文件解压到一个目录内，如 E:\Web，设置 E:\Web 为该新闻系统的网站主目录(该目录下有 admin 目录和 data 目录)，如图 10-37 所示。

该网站的数据库文件为 data 子目录下的 funonews. mdb。我们使用 Navicat for MySQL 软件将该 Access 数据库转换为 MySQL 数据库(具体转换方法见 10. 1. 4 节中)，数据库名为 test。

下面打开数据库 test，可发现该数据库中共有 4 个表，分别是 Admin、Bigclass、News 和 SmallClass，其中 News 表存放了网站中的全部新闻，News 表中字段及含义如表 10-3 所示。

表 10-3　News 表中的字段及含义

字　段　名	字 段 含 义	数 据 类 型

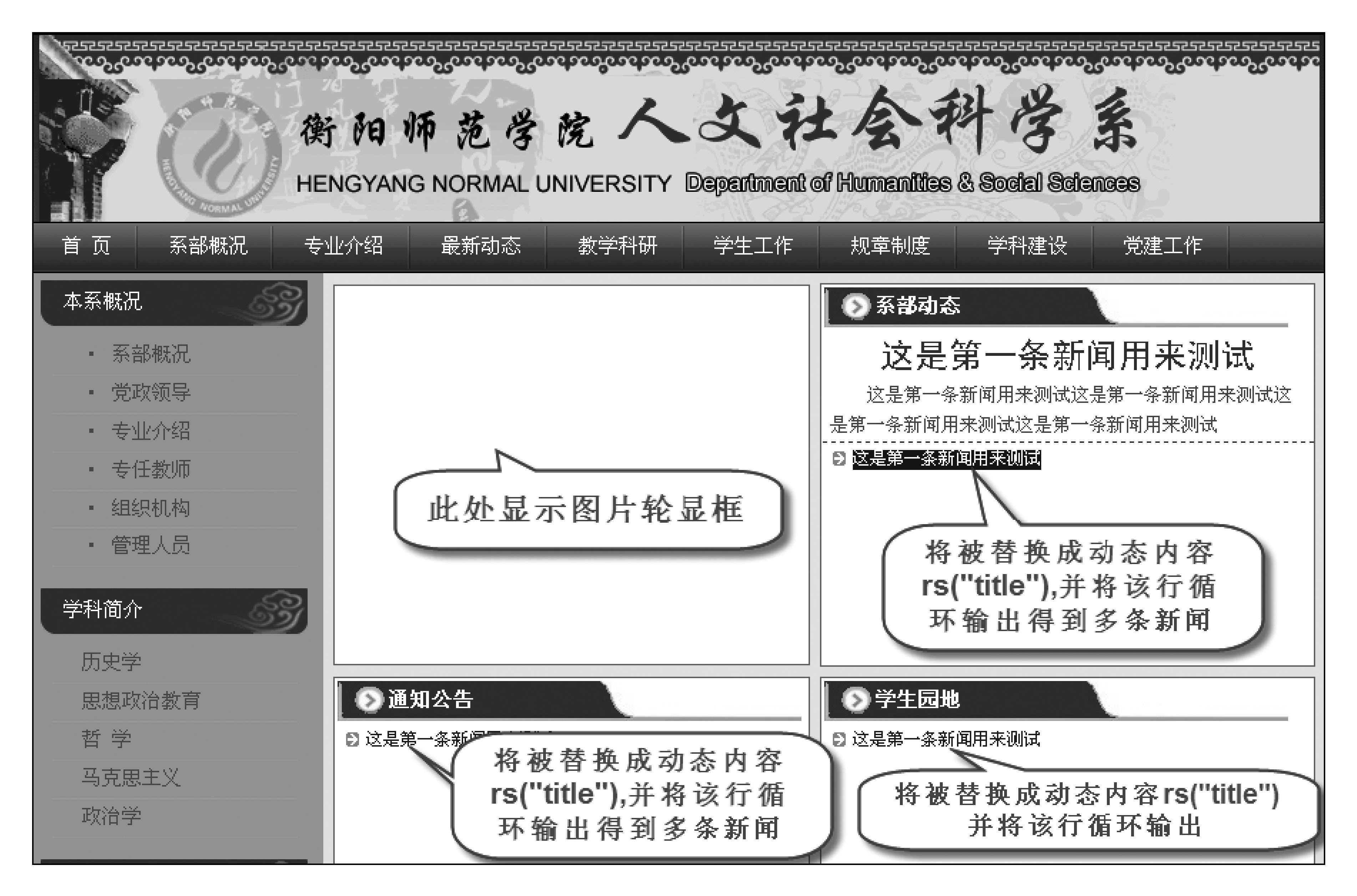


图 10-36 新闻网站的静态首页

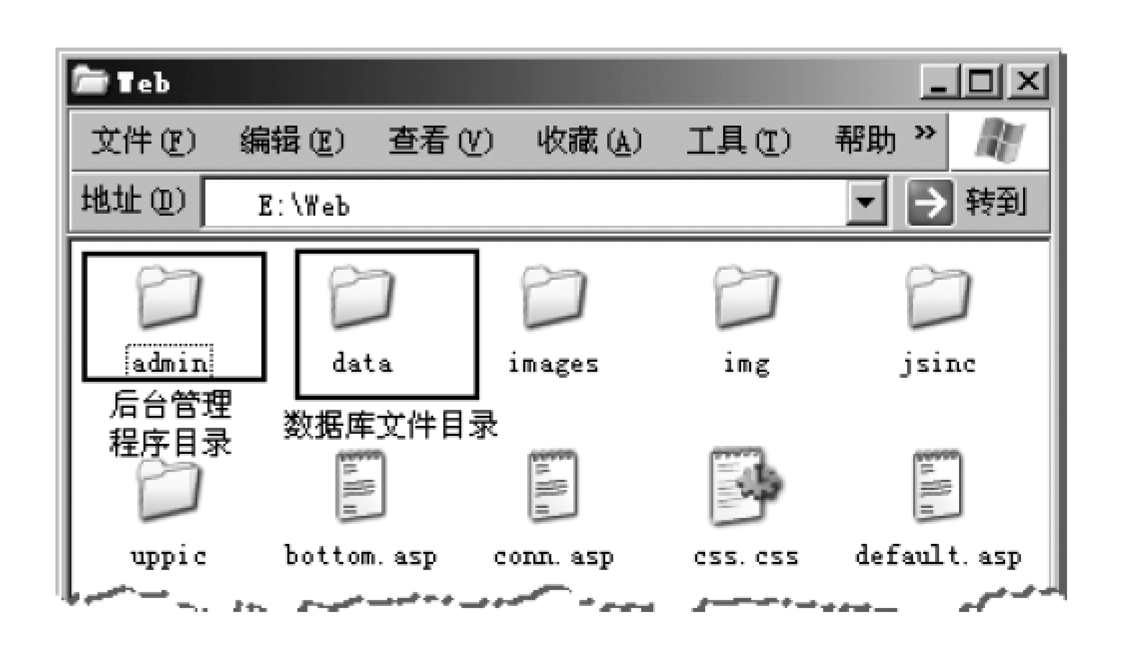


图 10-37 风诺新闻系统网站主目录下的内容

ID	新闻的编号	int,自动递增,主键
title	新闻标题	varchar
content	新闻内容	TEXT
BigClassName	新闻所属的大类名	varchar

续表

字　段　名	字 段 含 义	数 据 类 型
SmallClassName	新闻所属的小类名(可不指定)	varchar
imagenum	该条新闻中含的图片数	int
firstImageName	新闻中第一张图片的文件名	varchar
user	新闻发布者	varchar
infotime	新闻的发布日期	datetime
hits	该条新闻的点击次数	int
ok	是否将该新闻作为图片新闻显示(该新闻中必须含有图片)	tinyint

说明：新闻系统中的所有新闻是按栏目分类的。因此每条新闻中必须有一个 BigClassName 字段，以标注该新闻属于哪个栏目。一个新闻网站的结构及其对应页面如图 10-38 所示。

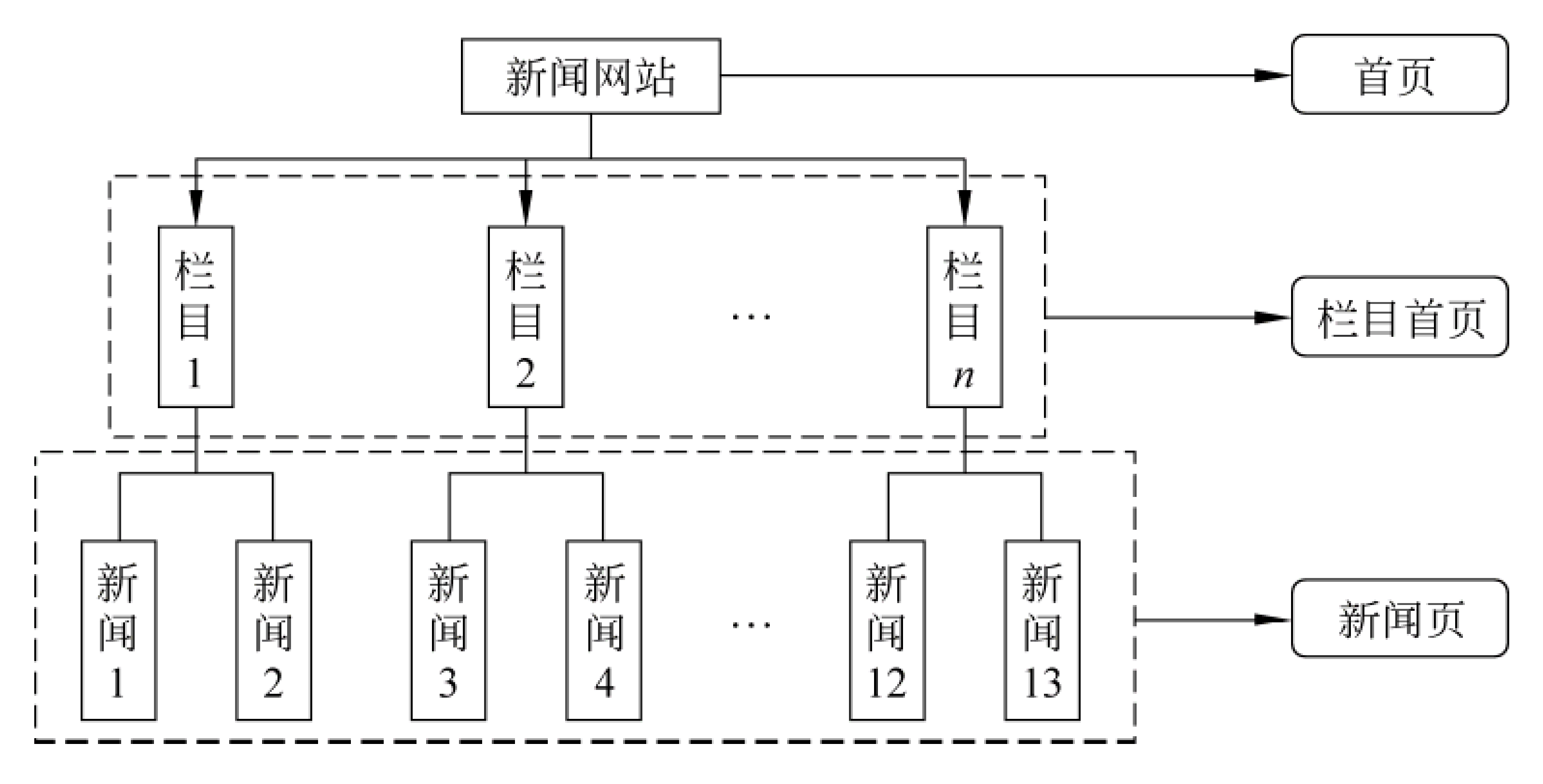


图 10-38　新闻网站的结构及其对应页面

网站目录下的其他文件是用该 CMS 制作的一个示例网站，其中 default. php 为该网站的首页，otype. php 为该网站的栏目首页，funonews. php 为该网站的内页。css. css 为该网站的样式表文件。top. php、bottom. php 和 left. php 为该网站各页面调用的头部、尾部和左侧文件。我们可以将这些文件都删除，只保留 admin 子目录(后台管理系统所在目录)。

接下来，进入风诺新闻系统的后台创建我们网站的栏目，并在每个栏目中添加几条新闻。后台登录的网址是 http://localhost/admin/adminlogin. php，使用默认用户名 funo 和密码 funo 即可登录进入如图 10-39 所示的新闻后台管理界面。

在这里，首先选择“管理新闻类别”创建网站应具有的栏目，如“通知公告”、“系部动态”、“学生园地”等，还可以在这些栏目下再选择“添加二级分类”来创建小栏目。将网站栏目创建好之后，就可以选择左侧的“添加新闻内容”为每个栏目添加几条测试新闻，只要在添加新闻时将这些新闻的“新闻类别”选择为不同的栏目即可。这样这些新闻就保存到了 news 表中。



图 10-39 网站后台管理界面

10.7.2 在首页显示数据表中的新闻

为了在网页上显示数据库中的新闻,必须先要连接数据库,本系统采用 mysqli 函数连接数据库,将连接数据库的代码写在 conn.php 中,以供其他页面调用。代码如下:

```
<? $conn=new mysqli();
$conn->connect('localhost','root','111');
$conn->select_db('test');
$conn->query('set names gb2312');    ?>
```

在首页的各个栏目中显示这个栏目的新闻是通过显示结果集中记录实现的。比如,要显示“通知公告”栏目中的最新 6 条新闻,可以执行下面的查询来创建结果集。

```
$result=$conn->query("select * from news where Bigclassname='通知公告' order by ID desc limit 6");
```

接下来就可以循环输出该结果集中的 6 条新闻到页面。而要显示“学生工作”栏目的新闻,就必须执行一个不同的查询,因此需要一个新的结果集来实现。为了得到一个新结果集,有两种方法:一种是创建一个新的结果集对象如 $result2;另一种方法是将原来的结果集关闭,再用原来的结果集对象 $result 打开一个新结果集。例如:

```
<? $result->close();     //关闭语句也可省略,mysqli 函数会自动关闭
$result=$conn->query("select * from news where Bigclassname='学生工作' order by ID desc limit 6");
    ?>
```

使用第二种方法内存中只需保存一个结果集对象,更节约资源,因此推荐使用。那么在首页显示新闻的过程是:

(1) 为第一个栏目创建结果集,然后循环输出结果集中记录到该栏目框中;

(2) 为第二个栏目创建结果集,再循环输出记录到第二个栏目框,如此循环。

因此,首页上有几个栏目就创建几个结果集,代码如下,运行结果如图 10-40 所示。

```
<div id="main">
```

```
<div id="pic">
    <? require('conn.php');
     include('flasha.php');                //flasha.php用来载入图片轮显框,具体代码见 10.7.3节
    ?></div>
<div id="xbdt">
<h2 class="lanmu"><a href="otype.php?owen1=近期工作"></a>系部动态</h2>
<?
$result=$conn->query("select  * from news where bigclassname in('规章制度','学生工作','德育园
地','科研成果','近期工作','图片新闻') order by ID desc limit 7 ");
$row=$result->fetch_assoc();                 ?>
<!--将最新的一条新闻以 h3标题的形式突出显示在系部动态栏目上方-->
<h3 align="center"><?=Trimtit($row['title'],12)?></h3>
<p><a href="onews.php?id=<?=$row['ID']?>">
<?=Trimtit(strip_tags($row['content']),40) ?></a> [<?=noyear($row
['infotime'])?>)]</p>
<ul>        <?       //显示系部动态栏目框中的 6条新闻
for($i=0;$i<6;$i++) {
        $row=$result->fetch_assoc();?>
<li class="xinwen"><b style="float:right;"><?=noyear($row['infotime'])?>
</b>
<a href="onews.php?id=<?=$row['ID']?>"><?=Trimtit($row['title'],20)?></a>
</li>
<? }
    $result->close();
    $result=$conn->query("select  * from news where Bigclassname='近期工作' order by ID desc
limit 6");             //为通知公告栏目创建结果集   ?>
</ul></div>
<div id="tzgg">
<h2 class="lanmu"><a href="otype.php?owen1=近期工作"></a>通知公告</h2>
<ul><?
for($i=0;$i<6;$i++) {
    $row=$result->fetch_assoc();     ?>
<li class="xinwen"><b style="float:right;"><?=noyear($row['infotime'])?></b>
<a href="onews.php?id=<?=$row['ID']?>"><?=trimtit($row['title'],20)?></a>
</li>
<? }
    $result->close();
    $result=$conn->query("select  * from news where Bigclassname='学生工作' order by ID desc
limit 6");             //为学生工作栏目创建结果集         ?>
</ul></div>
<div id="xsyd">
<h2 class="lanmu"><a href="otype.php?owen1=学生工作"></a>学生工作</h2>
<ul><?
for($i=0;$i<6;$i++) {
```

```
$row=$result->fetch_assoc();?>
<li class="xinwen"><b style="float:right;"><?=noyear($row['infotime'])?></b>
<a href="onews.php?id=<?=$row['ID']?>"><?=trimtit($row['title'],20)?>
</a></li>
<? }
    $result->close();
    $result=$conn->query("select * from NEWS where firstImageName<>'' order by ID DESC limit 7");
                //为图片滚动栏目创建结果集        ?>
</ul></div>
```



图 10-40 新闻版块最终效果图

将图 10-40 与图 10-36 相比，可看出图 10-36 中显示静态文字的地方被替换成了输出动态数据，这称为绑定数据到页面。由于每条新闻位于一个
标记内，因此循环输出新闻的循环体是
…。

上述代码中还调用了 3 个函数，即裁剪字符串长度的 Trimtit()函数(见 8.2.2 节例 8.5)、去除 HTML 标记的 PHP 内置函数 strip_tags()和去除日期前年份的函数 noyear(str)，代码如下：

```
<? function noyear($str) {
      return substr($str,5,5);
}  ?>
```

为了使本网站中所有网页都能调用这些函数，应将这些函数代码写在 conn.php 文件中。

上述代码调用的 CSS 代码如下，主要是设置 4 个栏目框浮动和设置标题栏背景图片。

```
<style type="text/css">
```

```
#pic,#xbdt,#tzgg,#xsyd {                    /* 4个栏目框 */
    border:1px solid #CC6600;
    background:white;
    width:335px;
    padding:2px 6px 10px;
    margin:4px;
    float:left;                             /* 使 4个栏目框都浮动 */}
#main {
    background:#e8eadd;
    padding:4px;}
#main .lanmu {
    background:url(images/title-bg3.jpg) no-repeat 2px 2px;
                                            /* 设置栏目标题背景图案 */
    padding:8px 0px 0px 40px;
    font-size:14px;
    color:white;
    margin:0;
    height:32px;}
#main .lanmu a {
    background:url(images/more2.gif) no-repeat;
                                            /* 设置超链接的背景为"more"图标 */
    float:right;                            /* 设置"more"图标右浮动 */
    width:37px;
    height:13px;
    margin-right:4px; }
#main #xbdt h3 {
    font: 24px "黑体";
    color:#900;                             /* 设置首条新闻的标题样式 */
    margin:0px 4px 4px;   }
#main #xbdt p {
    margin:4px;
    font: 13px/1.6 "宋体";
    color:#06C;
    text-indent:2em;
    border-bottom: 1px dashed #900;         /* 设置首条新闻与下面新闻的虚线 */}
#main .xinwen {
    height:24px;
    line-height:24px;
    background:url(images/article_common.gif) no-repeat 6px 4px;
                                            /* 新闻前的小图标 */
    font-size:12px;
    padding:0 6px 0 22px;}
.xinwen b {  float: right;                  /* 每条新闻的日期显示在右侧 */  }
ul{  margin:0;     padding:0;     list-style:none;   }
```

```
a {  color: #333;     text-decoration: none;   }
a:hover {   color: #900;   }
</style>
```

10.7.3 制作动态图片轮显效果

在图 10-40 中，第一个栏目框中图片轮显效果是通过包含一个 flasha.php 的文件实现的。该文件是通过调用一个 pixviewer.swf 的文件来实现图片轮显功能的。

1. pixviewer.swf 文件的原理

pixviewer.swf 是个特殊的 Flash 文件，用来实现图片轮显框。它可以接受两组参数。第一组参数包括 pics、links 和 texts，用于设置轮显图片的 URL 地址、图片的链接地址及图片下的说明文字。例如：

```
var pics="uppic/1.gif | uppic/2.gif | uppic/3.gif | uppic/4.gif | uppic/5.gif"
var links="onews.php?id=88 | onews.php?id=87 | onews.php?id=86 | onews.php?id=8 | onews.php?id=7"
var texts="爱我雁城、爱我师院 | 国培计划 | 青春舞动 | 长春花志愿者协会 | 朝花夕拾,似水流年"
```

这 3 个参数的值都是字符串，其中 pics 参数指定了欲载入图片的 url，这里使用了相对 url，共设置了 5 个图片文件的路径(最多可设置 6 个)。各图片路径之间必须用"|"号隔开(最后一幅图片后不能有"|")。links 参数定义了单击图片时的链接地址，texts 参数保存了每张图片下的说明文字，其格式要求和 pics 参数相同。上述代码载入了 5 幅图片轮显并定义了它们的链接地址和说明文字。

2. 轮显动态图片的方法

上述将 5 张图片 URL 地址直接写在 pics 变量中的做法只能固定地显示这 5 张图片。而在新闻网站中，通常要能自动显示最新的 5 条新闻中的图片。为此，必须能从 News 表中读取最新的 5 条具有图片的新闻记录，将记录的相关字段值填充到这 3 个参数中去。因为 News 表中的 firstImageName 字段保存了新闻中第一张图片的文件名，而这些新闻中的图片都保存在 uppic 目录中，因此可以采用如下语句为 pics 添加每幅图片的 URL 路径。

```
$pics.="uppic/".$row['firstImageName']."|";
```

而本新闻系统中所有的新闻都是链接到同一页面 onews.php，只是所带的参数为该条新闻的 id 字段。因此设置 links 参数的语句如下：

```
$links.="onews.php?id=".$row['ID']."|";
```

texts 参数只要装载每条新闻的标题即可，但要把标题长度限制在 16 个字符以内。

```
$texts.=Trimtit($row['title'],16)."|";
```

下面是从数据库中读取 5 条具有图片的记录，并设置 pics、links、texts 参数，实现轮

显动态图片的代码(flasha.php):

```
<script>
   var pics="",links="",texts="";
<?          //用 Select 语句查询 5 条最新的新闻且新闻中图片不为空
$sql="select * from news where firstImageName<>'' and ok=true order by ID desc limit 5";
$result=$conn->query($sql);
while($row=$result->fetch_assoc()){
     $pics.="uppic/".$row['firstImageName']."|";
        $links.="onews.php?id=".$row['ID']."|";
        $texts.=Trimtit($row['title'],16)."|";
}
        $pics=substr($pics,0,-1);        //去除最后一条记录后的"|"
        $links=substr($links,0,-1);
        $texts=substr($texts,0,-1);     ?>
     var pics="<?=$pics ?>";
     var links="<?=$links ?>";
     var texts="<?=$texts ?>";
…
</script>
```

说明:创建结果集时选择了图片不为空且允许作为图片新闻显示的 5 条记录。在输出记录时,记录之间必须添加分隔符"|",但最后一条记录之后不能有分隔符"|",为此,通过 substr 函数将最后一条记录后的"|"去除。

3. 设置图片轮显框的大小

第二组参数用来定义该图片轮显框及其说明文字的大小。它有 4 个参数,包括:

```
var focus_width=336                       //定义图片轮显框的宽
var focus_height=224                      //定义图片轮显框的高
var text_height=14                        //定义下面文字区域的高
var swf_height=focus_height+text_height          //定义整个 FLash 的高
```

只要修改这些参数,就能使图片轮显框改变成任意大小显示。

4. 其他设置

下面还有一些代码,用来将 Pixviewer.swf 这个 Flash 文件插入到网页中,并对其设置参数的代码。这段代码不需要做多少修改,只要保证引用 Pixviewer.swf 文件的 URL 路径正确,还可以设定文字部分的背景颜色。找到第二个 document.write,粗体字为设置的地方。

```
document.write('<param name="allowScriptAccess" value="sameDomain"><param name="movie" value="
images/pixviewer.swf"><param name="quality" value="high"><param name="bgcolor" value="#ffffff">
');
```

10.7.4 制作新闻内容页

新闻内容页实际上就是显示一条记录的页面，它首先获取前一页面传过来的记录id，找到id对应的新闻后，将要显示的字段用不同的样式输出到页面的对应位置上，如图10-41所示。例如，title字段以24px红色字体显示在页面上方，而content字段以正常字体显示在页面中央。



图10-41 显示新闻详细页面

1. 显示新闻的制作

新闻详细页面首先应使用$_GET['id']获取其他页超链接中传过来的id参数，再根据该id构造Select语句找到这条新闻，然后将新闻中的各个字段存入到数组$row中。代码如下：

```
<? require('conn.php');
$id=intval($_GET['id']);
$sql="select * from news where id=$id";
$result=$conn->query($sql);          //根据记录id创建结果集
if($result->num_rows>0){
$row=$result->fetch_assoc();     …}     ?>
```

接下来，就可以将该条记录的各个字段输出到页面的相应位置，主要代码如下：

```
<title><?=$row['title'] ?></title>     <!--将title字段显示在页面标题中-->
…<h1><?=$row['title'] ?></h1>
当前位置:<a href="index.php">首页</a>&gt;<a href="otype.php?owen1=<?=$row['bigclassname'] ?
>"><?=$row['bigclassname'] ?>
发布者:<?=$row['user'] ?>发布时间:<?=notime($row['infotime'])?>阅读:<font color="#ffcc00">
<?=$row['hits'] ?></font>次
<div style='font-size:7.5pt'>
<hr width="700" size="1" color="cccc99">
<?=$row['content'] ?></div>
```

显示完结果集后，必须将结果集和数据库连接关闭，否则可能会影响网站内其他页面

的打开速度,关闭结果集和数据库连接的代码如下:

```
<? $result->close();
   $conn->close();    ?>
```

2. “上一条”“下一条”新闻链接的制作

在显示新闻页面中,“上一条”链接可以链接到该新闻所属栏目中的上一条新闻;“下一条”链接则转到同栏目中的下一条新闻,如图 10-41 所示。虽然这种功能对于新闻网站来说并不是十分必要,但对于博客类网站来说却是不可或缺的,因为我们通常都是通过单击“下一条”链接来一条条查看博客主人的日志。

制作的思路如下:“上一条”链接是要找到本栏目中上一条新闻的 id 值。这不能通过将本条新闻的 id 值减 1 实现,因为这样得到的 id 值对应的新闻可能是其他栏目的新闻,甚至可能是已经被删除了的新闻(删除记录后其 id 值不会被新添加的记录所占用)。而应该通过一个查询语句,找到在同一栏目(bigclassname)中所有 id 值比该新闻的 id 值小的记录,再对这些记录进行逆序排列,取其中 id 值最大的一条,也就是逆序排列后结果集中的第一条记录。

因此,首先要通过 id 找到该条记录对应的大类名(bigclassname),将其保存到变量 $ bcn 中,然后关闭结果集,再新开一个查询上一条新闻 id 和 title 的结果集。代码如下:

```
<? require('conn.php');
$id=intval($_GET['id']);
$sql="select bigclassname from news where ID=$id ";                //根据记录 id找到大类名
$result=$conn->query($sql);
$row=$result->fetch_row();
$bcn=$row[0];
              //找到该大类中与该记录相邻的上一条记录
$sql="select ID,title from news where ID<$id and Bigclassname='$bcn' order by ID desc limit 1";
$result=$conn->query($sql);
if($result->num_rows==0)       //如果结果集为空
    $pret=0;                   //令 $pret 为 0表示上一条记录没有了
else {
    $row=$result->fetch_assoc();
    $pret=$row['ID'];
    $pretit=$row['title'];     }
$sql="select ID,title from news where ID>$id and Bigclassname='$bcn' order by ID limit 1";
$result=$conn->query($sql);
if($result->num_rows==0)
    $nextt=0;                  //令 $nextt 为 0表示下一条记录没有了
else {
    $row=$result->fetch_assoc();
    $nextt=$row['ID'];
    $nexttit=$row['title'];    }
?>
```

接下来,在页面上输出“上一条”“下一条”及“当前位置”的链接,代码如下:

```
<?    if($nextt<>0){          //如果有下一条记录    ?>
    <a title="<?=$nexttit ?>" style="float:right;padding-right:16px;" href="onews.php?id=<?=$nextt ?>">下一条 &gt;&gt;</a>
<?}
    if($pret<>0) {            //如果有上一条记录    ?>
    <a title="<?=$pretit ?>" style="float:right;padding-right:16px;" href="onews.php?id=<?=$pret ?>">&lt;&lt; 上一条</a>
<?}?>
    当前位置:<a href="index.php">首页</a> &gt;
    <a href="otype.php?owen1=<?=$bcn ?>"><?=$bcn ?></a>
```

3. 记录新闻的单击次数

只要将下面的语句放在页面的适当位置,用户每打开一次该页面,就会使 hits 值加 1。

```
<?  $sql="update news set hits=hits+1 where id=$id";
$conn->query($sql);    ?>
```

10.7.5 制作栏目列表页

栏目列表页用来显示某个栏目中的新闻,如图 10-42 所示。当用户单击导航条上的某个导航项或栏目框上的 more 图标时,都将链接到栏目列表页,并将栏目名以 URL 参数的形式传递给该页。因此,栏目列表页首先要获取栏目名。

在网页左侧根据栏目名显示子栏目列表的代码如下:

```
<?  $owen1=$_GET["owen1"];          //获取首页传来的一级栏目名
      $owen2=$_GET["owen2"];        //获取首页传来的子栏目名
    //根据一级栏目名显示下面的子栏目名
$result=$conn->query("Select * From SmallClass Where BigClassName='$owen1'");
if($result->num_rows>0){
while($row=$result->fetch_assoc()){     ?>
      <tr bgColor=#EFEFEF>              <!--一行显示一个子栏目名-->
      <td height="25" align="center" bgcolor="#EFEAD8">
<a  href="otype.php?owen1=<?=$owen1 ?>&owen2=<?=$row["SmallClassName"] ?>">
<b><?=$row["SmallClassName"] ?></b></a></td></tr>
<?    }}    ?>
```

在网页右侧根据栏目名执行查询得到该栏目新闻记录的列表并显示。代码如下:

```
<?    if($owen1<>"" &&$owen2<>"")          //如果获取的一级栏目和子栏目名都非空
$sql="select * from news where BigClassName='$owen1' and SmallClassName=
'$owen2' order by ID desc";
```

```
else if($owen1<>"")      //如果获取的一级栏目名不为空,则根据一级栏目名查询
$sql="select * from news where BigClassName='$owen1' order by ID desc";
$result=$conn->query($sql);
$RecordCount=$result->num_rows;      ?>
```

接下来就是将该结果集所有记录的标题和日期等字段循环输出到页面上。

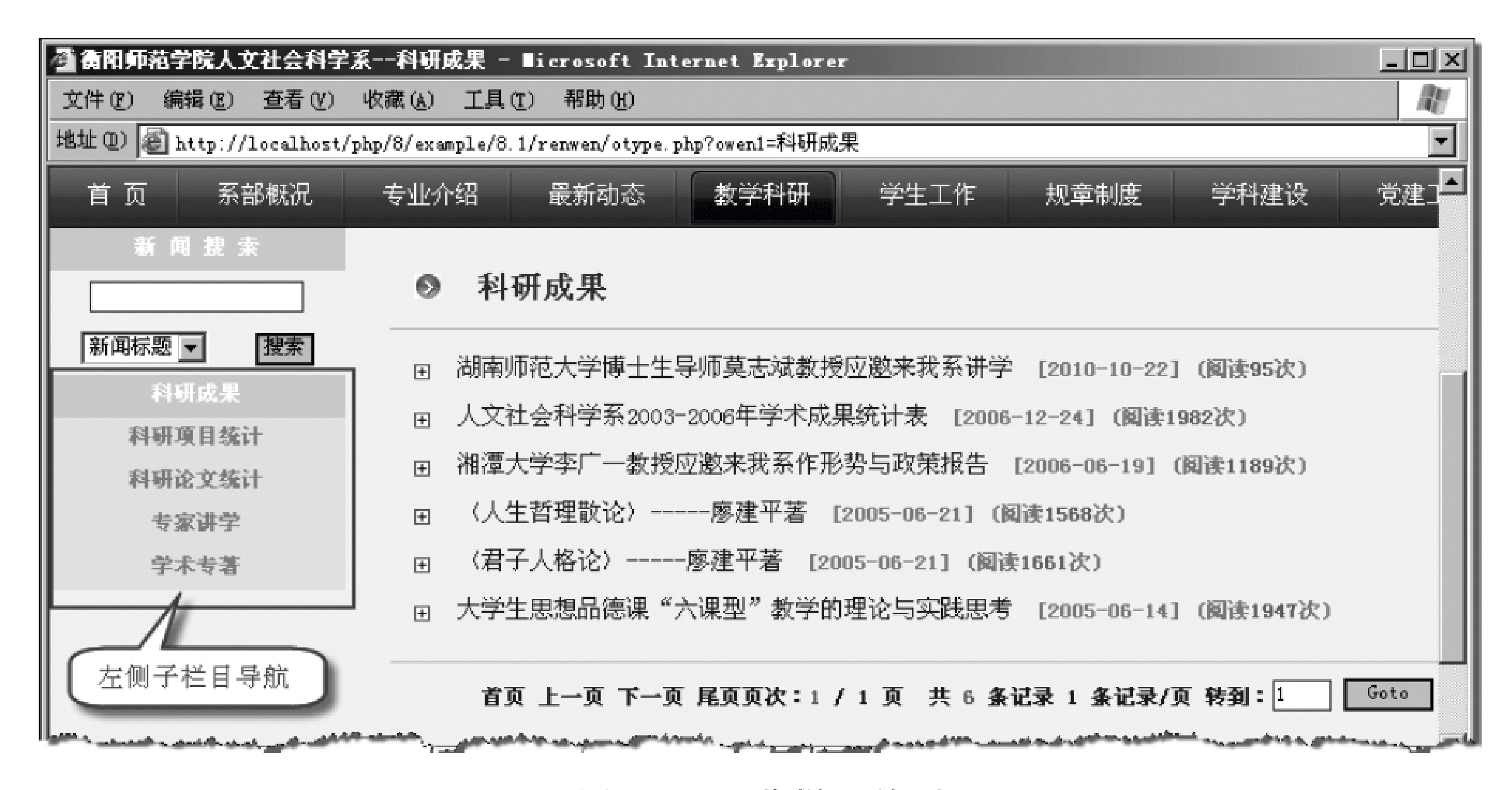


图 10-42　分栏目首页

由于每个栏目的记录可能有很多,因此图 10-42 中的分栏目首页还应具有分页功能,分页功能的实现请读者仿照 10.5.2 节中介绍的方法实现。

10.7.6　使用 FCKeditor 编辑器

我们编辑新闻时,如果用表单中的多行文本域,则只能在其中输入纯文本,如果要对这些文本进行网页排版则很不方便。为此,人们开发了在线编辑器软件,这些在线编辑器就像 Dreamweaver 的设计视图,可以对新闻中的文字和图片进行可视化排版,使新闻以美观、适合阅读的版式显示出来。

FCKeditor 是目前流行的"所见即所得"的在线编辑器,它具有功能强大、体积小巧、跨浏览器、支持多种 Web 编程语言等特点。本节以 FCKeditor 2.6.8 版本为例,结合新闻发布系统讲解 FCKeditor 的使用。

在百度上搜索"FCKeditor 下载",将下载的压缩文件 FCKeditor_2.6.8 解压到新闻发布系统的根目录下即可,FCKeditor 的目录结构如图 10-43 所示。

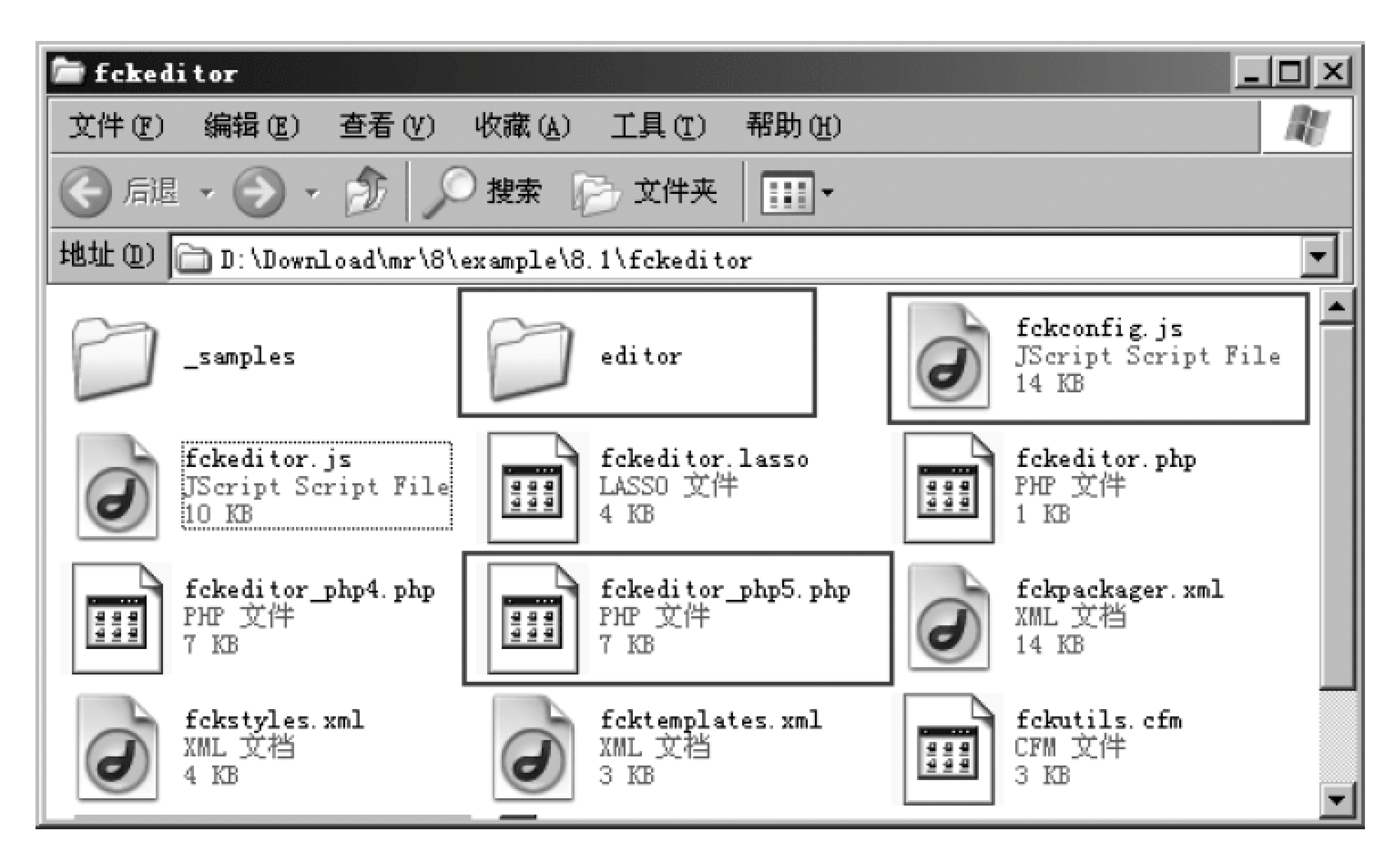


图 10-43 FCKeditor 的目录结构

fckeditor 文件夹下比较重要的目录和文件有：

(1) editor 目录定义了编辑器的 CSS 样式表、皮肤文件、图片以及文件上传程序等文件。

(2) _samples 目录提供了 FCKeditor 的示例程序。

(3) fckeditor_php5. php 是 PHP 程序实例化 FCKeditor 的类文件。

(4) fckconfig. js 是 FCKeditor 工具栏集合的配置文件，这两个文件对于我们配置和使用该编辑器具有至关重要的作用。

1. 调用 fckeditor 编辑器

fckeditor_php5. php 代码中定义了一个 FCKeditor 类，调用 FCKeditor 编辑器必须先用该类创建一个实例化对象。而要创建 FCKeditor 类的实例，必须先载入 FCKeditor 类文件。例如：

```
<?   include("fckeditor/fckeditor_php5.php");      //载入 FCKeditor 类文件
     $oFCKeditor=new FCKeditor('content');          //创建 FCKeditor 类的实例
?>
```

接下来，设置 FCKeditor 实例的根目录以及 FCKeditor 实例其他成员属性的值，最后用 Create()方法创建编辑器，以 10.4.5 节中新闻编辑页面 editform. php 为例，将该程序中多行文本框代码：

```
<textarea name="content" cols="30" rows="2"><?=$row['content'];?></textarea>
```

替换为：

```
<?  include("fckeditor/fckeditor_php5.php");       //载入 FCKeditor 类文件
$oFCKeditor=new FCKeditor('content');               //创建 FCKeditor 类的实例
$oFCKeditor->BasePath='fckeditor/';                 //设置 FCKeditor 目录地址为
//当前目录下的 fckeditor 目录,这样 FCKeditor 才能找到它的相关资源文件
$oFCKeditor->Width='95%';                          //设置显示宽度
```

```
$oFCKeditor->Height='400px';                //设置显示高度
$oFCKeditor->Value=$row['content'];         //设置编辑器的值,将显示在编辑器中
$oFCKeditor->Create();                      //创建编辑器
?>
```

则通过更新链接进入 editform.php 页面的运行结果如图 10-44 所示。



图 10-44 将 FCKeditor 编辑器嵌入到 editform.php 页面中

说明:一个 FCKeditor 编辑器实例和普通的表单控件一样,也具有 name 属性和 value 属性,上例中编辑器的 name 属性值为 content,是通过 new FCKeditor('content')设置的,value 属性值为 $ row['content'],是通过 $ oFCKeditor->Value 属性来设置的。因此,要获取该编辑器中的内容,可以用 $ content= $ _POST["content"]来获取;要在该编辑器中显示内容,可以通过"$ oFCKeditor->Value='要显示的内容';"语句来实现。

对于 10.4.2 节中新闻添加页面 addform.php,也可以使用上述方法将多行文本域替换成 FCKeditor 编辑器,只是因为新闻添加页面中的新闻内容为空,因此不需要设置 $ oFCKeditor->Value 属性的值。

2. 配置 fckeditor 编辑器的文件上传功能

FCKeditor 编辑器内置了文件管理功能,使得用户能够上传图像或文件,它的文件管理程序放在 editor/filemanager/目录下,FCKeditor 提供了文件浏览和文件快速上传功能,"文件浏览"为用户提供了 3 种功能:浏览服务器上已存在的多媒体文件、在编辑器中浏览多媒体文件以及上传本地文件到服务器。"快速文件上传"为用户提供了快速上传本地文件至服务器的功能,是"文件浏览"功能的子集。

为了能够使用快速文件上传功能,需要进行一些配置,打开 fckeditor\editor\filemanager\connectors\php 目录中的 config 文件,找到如下两行代码:

```
$Config['Enabled']=true;                    //将 false 改为 true,表示允许上传
$Config['UserFilesPath']='upfiles/';        //定义上传目录
```

再打开 fckeditor 根目录下的 fckeditor.js,确保以下两行的值为 php:

```
var _FileBrowserLanguage    ='php';
```

```
var _QuickUploadLanguage   = 'php';
```

至此,Apache 服务器已能正常上传文件,但对于 IIS 服务器,还需要设置上传目录的绝对路径,找到类似下面的代码将其修改为:

```
$Config['UserFilesAbsolutePath']= 'D:\\AppServ\\www\\upfiles';
```

3. 配置对上传文件进行重命名

如果要上传的文件文件名中存在中文字符,则会出现乱码问题,导致引用的文件URL 不对,解决这个问题的办法是将所有上传文件的文件名重命名,方法如下:

找到 editor\filemanager\connectors\php 目录下的 io. php 文件,将函数名为SanitizeFolderName 的函数代码修改如下:

```
function SanitizeFileName($sNewFileName)    {
    $arr= explode('.',$sNewFileName);
    $ext= array_pop($arr);     //第一个数组元素保存了.前的文件名
    $filename= date('Ymd_His_').rand(1000,9999).'.'.$ext;
    return $filename;
}
```

SanitizeFolderName 函数的功能是将上传文件的文件名重命名,新的文件名按照日期时间随机数的格式命名,既可防止修改后的文件名重名,又可防止新的文件名中出现中文字符。

4. 解决文件上传功能的安全性问题

文件上传可能给网站带来巨大的安全隐患,假设网站攻击者猜测到了文件上传程序的路径,则他可以直接访问该程序,以上传文件。为此,需要判断上传文件者是否是登录成功的用户,这可以通过 Session 变量判断。假如网站采用 $_SESSION['admin']验证管理员是否已经登录,则只需打开 config. php 文件。将配置上传的代码修改为:

```
$Config['Enabled']= isset($_SESSION['admin']);
```

这样,当用户未登录时,isset 函数将返回 false。

10.8 数据库接口层 PDO

PHP 提供了操作各种数据库的内置函数,通过这些内置函数 PHP 可直接访问数据库。例如,使用 mysql 或 mysqli 函数库能够直接访问 MySQL 数据库,使用 mssql 函数库能直接访问 SQL Server 数据库。而如果要访问 Oracle 数据库,就需要使用 ora 函数(或oci 数据抽象层)。可见,访问每种数据库时都需要学习特定的函数库,这是比较麻烦的。更重要的是,如果要将 PHP 程序移植到其他数据库上,就需要修改大量的程序代码,使移植难以实现。

为了解决这个问题,出现了“数据库接口层”的概念。通过数据库接口层可以访问任

意类型的数据库，而 PHP 程序只要与接口层打交道，发送统一的指令给这个通用接口，再由接口层将指令传输给任意类型的数据库，如图 10-45 所示。

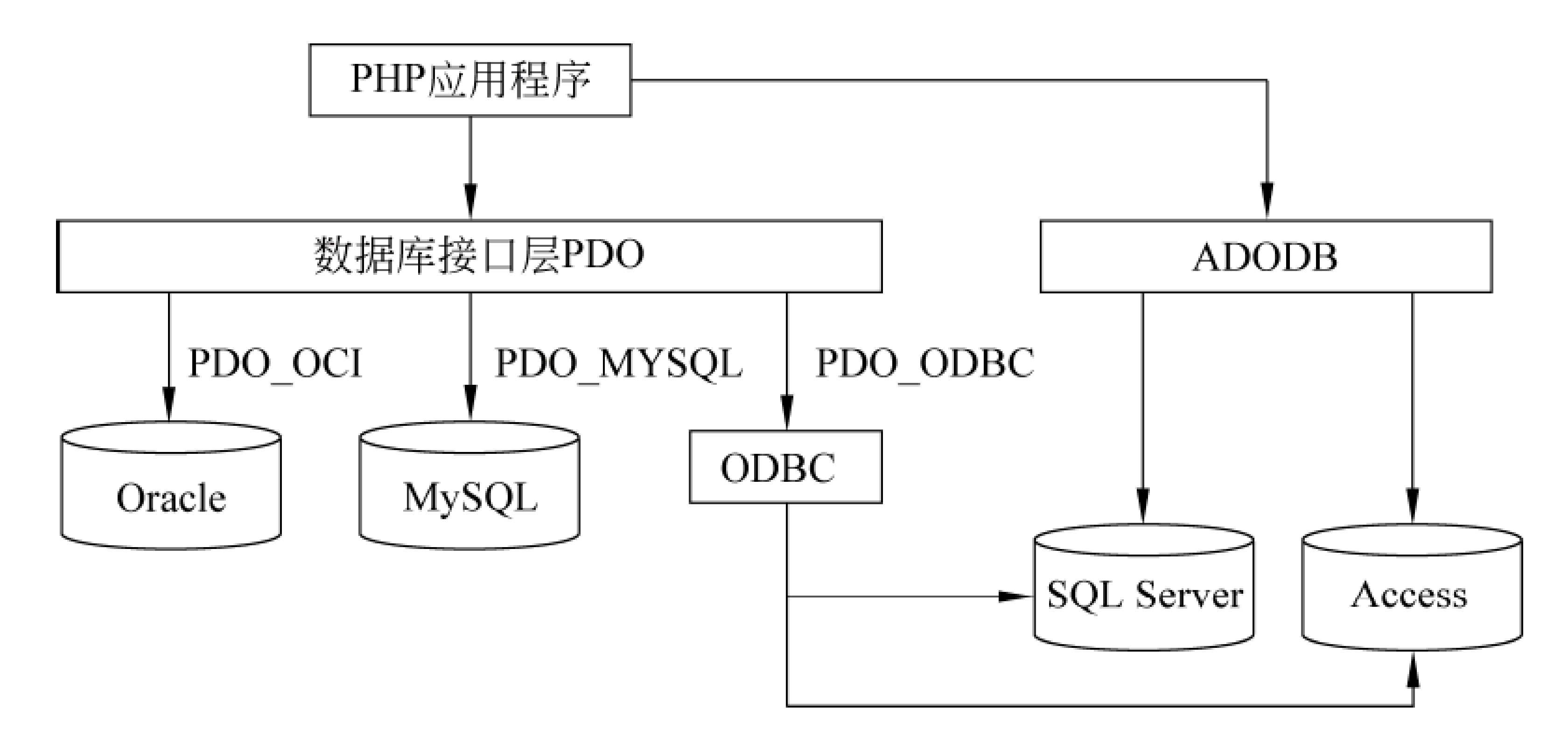


图 10-45　数据库访问接口层

PDO(PHP Data Object)是为 PHP 访问数据库定义的一个轻量级的、一致性的数据库接口，它提供了一个数据库访问抽象层，作用是统一各种数据库的访问接口，使得程序能够轻松在不同数据库之间进行切换，使数据库间的移植变得容易实现。这样，无论使用什么数据库，都可以通过一致的函数执行查询和获取数据。

提示：PDO 是 PHP 5 新加入的一个重大功能，并且 PHP 6 将默认使用 PDO 来操作数据库，可见 PDO 是将来 PHP 在数据库处理方面的主要发展方向。

常见的数据库接口层除了 PDO 外，还有 ADO(ActiveX Data Object)，ADO 是微软推出的，一般用来访问微软的数据库，如 SQL Server 或 Access。而 PDO 一般用来让 PHP 访问非微软的数据库，如果一定要用 PDO 来访问微软的数据库，那么可以使用它提供的 PDO_ODBC 驱动连接 ODBC，再通过 ODBC 访问微软的数据库。

10.8.1　PDO 的安装

安装 PHP 5.1 以上版本会默认安装 PDO，但使用之前，仍需要进行一些相关的配置。打开 PHP 的配置文件 php.ini，在其中的 Dynamic Extensions 一节找到：

```
;extension=php_pdo.dll
```

将前面的“;”号(注释符)去掉，就打开了 PDO 所有驱动程序共享的扩展。接下来，还需要激活一种或多种 PDO 驱动程序，添加下面的一行或多行代码即可。

```
extension=php_pdo_mysql.dll          //如果要使用 MySQL,那么添加这一行
extension=php_pdo_mssql.dll          //如果要使用 SQL Server
extension=php_pdo_oci.dll            //如果要使用 Oracle
extension=php_pdo_odbc.dll           //如果要使用 ODBC 驱动程序
```

保存修改后的 php.ini 文件，然后重启 Apache 服务器，即完成了 PDO 的安装。

10.8.2　创建 PDO 对象连接数据库

在使用 PDO 与数据库交互之前，必须先创建一个 PDO 对象。创建 PDO 对象有多种

方法，其中最简单的一种方法如下：

```
对象名=new PDO(string DSN, string username, string password, [array driver_options]);
```

说明：

(1) 第一个必选参数是数据源名(DSN)，用来指定一个要连接的数据库和连接使用的驱动程序。其语法格式为：

```
驱动程序名:参数名=参数值;参数名=参数值
```

例如，连接 MySQL 数据库和连接 Oracle 数据库的 DSN 格式分别如下：

```
mysql:host=localhost;dbname=testdb
oci:dbname=//localhost:1521/mydb
```

(2) 第二个参数和第三个参数分别用于指定连接数据库的用户名和密码，是可选参数。

(3) 第四个参数 driver_options 必须是一个数组，用来指定连接所需的所有额外选项，传递附加的调优参数到 PDO 底层驱动程序。

下面是一个创建 PDO 对象并连接 MySQL 数据库 guestbook 的代码(conn.php)：

```
----------清单 10-19 conn.php    使用 PDO 对象连接数据库-----------
<? $dsn="mysql:host=localhost;dbname=guestbook";
   $db=new PDO($dsn,'root','111');          //连接数据库
   $db->query('set names gb2312');          //设置字符集
?>
```

提示：上述创建的连接数据库代码默认不是长连接，如果需要数据库长连接，需要用到第四个参数：array(PDO::ATTR_PERSISTENT=> true)，即：

```
$db=new PDO($dsn,'root','111', array(PDO::ATTR_PERSISTENT=>true));
```

当 PDO 对象创建成功后，与数据库的连接已经建立，就可以使用该对象了。PDO 对象中常用的成员方法如表 10-4 所示。

表 10-4　PDO 类中常用的成员方法

方　法　名	描　　述
query()	执行一条有结果集返回的 SQL 语句，并返回一个结果集 PDOStatement 对象
exec()	执行一条 SQL 语句，并返回所影响的记录数
lastInsertId()	获取最近一条插入到表中记录的自增 id 值
prepare()	负责准备要执行的 SQL 语句，用于执行存储过程等

调用 PDO 对象的方法可以使用“对象名->方法名”的形式。使用 PDO 对象的 query() 方法执行 Select 语句后会得到一个结果集对象 PDOStatement，该对象的常用方法如表 10-5 所示。

表 10-5　PDOStatement 类中常用的成员方法

方法名	描　　述
fetch()	以数组或对象的形式返回当前指针指向的记录，并将结果集指针移至下一行，当到达结果集末尾时返回 False
fetchAll()	返回结果集中所有的行，并赋给返回的二维数组，指针将指向结果集末尾
fetchColumn()	返回结果集中下一行某个列的值
setFetchMode()	设置 fetch()或 fetchAll()方法返回结果的模式，如关联数组、索引数组、混合数组、对象等
rowCount()	返回结果集中的记录总数，仅对 query()和 prepare()方法有效
columnCount()	在结果集中返回列的总数
bindColumn()	将一个列和一个指定的变量名绑定(必须设置 fetch 方法为 FETCH_BOTH)

10.8.3　使用 query()方法执行查询

PDO 访问数据库和 mysql 函数访问数据库的步骤基本上是一致的，即：

(1) 连接数据库；

(2) 设置字符集；

(3) 创建结果集；

(4) 读取一条记录到数组；

(5) 将数组元素显示在页面上。

使用 query()方法可以执行一条 select 查询语句，并返回一个结果集。例如：

```
$result=$db->query('select * from news limit 20');
```

也可使用 query()方法来设置字符集，但必须在创建结果集之前使用。例如：

```
$db->query('set names gb2312');
```

例 10.1　使用 query()执行查询的示例程序。

该程序将以表格的形式显示结果集中所有记录到页面上，运行结果如图 10-20 所示。

```
<?  $dsn="mysql:host=localhost;dbname=guestbook";
    $db=new PDO($dsn,'root','111');            //连接数据库
    $db->query('set names gb2312');            //设置字符集
    $result=$db->query('select * from lyb');   //执行查询创建结果集
    $result->setFetchMode(PDO::FETCH_ASSOC);
    //print_r($row=$result->fetch());
?>
<table border="1" width="95%">
  <tr bgcolor="#e0e0e0">
    <th>标题</th><th>内容</th><th>作者</th><th>email</th><th>来自</th></tr>
  <? while($row=$result->fetch()){              //读取一条记录到数组$row中
?>
```

```
  <tr><td><?=$row['title'];?></td><td><?=$row['content'];?></td>
    <td><?=$row['author'];?></td><td><?=$row['email'];?></td>
    <td><?=$row['ip'];?></td></tr>
  <? } ?>
</table><p>共有<?=$result->rowCount()?>行</p>
```

说明：

(1) 创建了结果集 \$result 后，可以用 \$result->fetch()方法读取当前记录到数组中，该数组默认是混合数组，如果希望 fetch()方法只返回关联数组，有两种方法：

① 在创建了结果集后用 \$result->setFetchMode(PDO::FETCH_ASSOC)方法进行设置。

② 给 fetch()方法添加参数，如 \$row= \$result->fetch(PDO::FETCH_ASSOC)或 \$row= \$result->fetch(1)。在 fetch()参数的可选值中，0 代表混合数组，默认值；1 或 2 代表关联数组，3 代表索引数组。本例中采用的是第①种方法。

(2) \$result->rowCount()方法可以返回结果集中的记录总数。

(3) 可以使用 print_r 方法打印 \$result->fetch(2)返回的数组，如果去掉本例中 print_r 语句前的注释符，就会输出：

```
Array([ID]=>1 [title]=>祝大家开心 [content]=>非常感谢大家长期以来的帮助 [author]=>唐三彩
[email]=>sanyo@tom.com [ip]=>59.51.24.37)
```

(4) 在 PDO 中使用 query()方法创建的结果集 \$result 是对象类型，var_dump (\$result)会得到：

```
object(PDOStatement)#2 (1) { ["queryString"]=>string(17) "select * from lyb" }
```

而以前用 mysql_query()创建的结果集 \$result 是资源类型，两种结果集的数据类型是不同的。

10.8.4 使用 fetchAll()方法返回所有行

fetchAll()方法将返回一个二维数组，该二维数组中包含了结果集中的所有行，并将结果集指针移动到结果集末尾。示例代码如下，运行效果如图 10-20 所示。

```
<?   require('conn.php');                        //conn.php 见 10.8.2 节清单 10-19
    $result=$db->query('select * from lyb');       //执行查询创建结果集
    $result->setFetchMode(PDO::FETCH_ASSOC);
?>
    <table border="1" width="95%">
      <tr bgcolor="#e0e0e0">
        <th>标题</th><th>内容</th><th>作者</th><th>email</th><th>来自</th>
  </tr>
  <? while($rows=$result->fetchAll()){   //读取所有记录到二维数组$rows 中
             foreach($rows as$row) {      //将每条记录放到数组$row 中
?>
```

```
    <tr><td><?=$row['title'];?></td><td><?=$row['content'];?></td>
      <td><?=$row['author'];?></td><td><?=$row['email'];?></td>
      <td><?=$row['ip'];?></td></tr>
    <? } }?>
</table><p>共有<?=$result->rowCount()?>行</p>
```

请注意：由于 fetchAll()返回的是二维数组，因此读取其中的所有字段需要使用双重循环。

对于结果集比较小时，用 fetchAll()方法效率较高，因为它可减少从结果集中移动指针的次数；但对于大结果集，一次读取所有记录会占用很大内存，此时使用 fetch()方法更合适。

10.8.5 使用 exec()方法执行增、删、改命令

如果要用 PDO 对数据库执行添加、删除、修改操作，则可以使用 exec()方法，该方法将处理一条 SQL 语句，并返回所影响的记录条数。

例 10.2 使用 exec()方法修改记录的示例程序。

```
<?  require('conn.php');              //conn.php见 10.6.2节清单 10-19
     $affected=$db->exec("update lyb set content='用 PDO修改记录' where author='蓉蓉'");
   ?>
<p>共有<?=$affected ?>行记录被修改</p>
```

如果要执行添加、删除操作，只要把上例中的 SQL 语句改为 insert 或 delete 语句即可。

读者可以使用 exec()方法改写 10.4 节中的所有程序，实现对记录的增、删、改操作。

10.8.6 使用 prepare()方法执行预处理语句

PDO 提供了对预处理语句的支持。预处理语句的作用是：编译一次，可以多次执行。它会在服务器上缓存查询的语法和执行过程，而只在服务器和客户端之间传输有变化的列值，以此来消除这些额外的开销。例如，要插入 1000 条记录，如果使用 exec 方法则需执行 1000 条 insert 语句，而使用预处理语句则只要编译执行一条插入语句。

对于复杂查询来说，如果要重复执行许多次有不同参数的但结构相同的查询，通过使用一个预处理语句就可以避免重复分析、编译、优化的环节。因此在执行重复的单个查询时快于直接使用 query()或 exec()方法，并且可以有效防止 SQL 注入(因为 SQL 语句是固定的，不需接收用户输入的参数值)，因此这种方法速度快而且安全。

执行预处理语句的过程是：

(1) 在 SQL 语句中添加占位符，PDO 支持两种占位符，即问号占位符和命名参数占位符，具体使用哪种凭个人喜好。两种占位符示例如下：

```
$sql="insert into lyb(title,content,author) values(?,?,?)";     //?号占位符
$sql="insert into lyb(title,content,author) values(:title,:content,
:author)";      //命名参数占位符
```

（2）使用 prepare()方法准备执行预处理语句，该方法将返回一个 PDOStatement 类对象。例如：

```
$stmt=$db->prepare($sql);
```

（3）绑定参数，使用 bindParam()方法将参数绑定到准备好的查询的占位符上。例如：

```
$stmt->bindParam(1,$title);          //对于?号占位符,绑定第 1个参数
$stmt->bindParam(':title',$title);   //对于命名参数占位符,绑定:title参数
```

（4）使用 execute()方法执行查询。例如：

```
$stmt->execute();
```

也可以在执行查询的同时绑定参数，execute 方法的参数是一个数组。代码如下：

```
$stmt->execute(array('PDO预处理','这是插入的记录','西贝乐'));
```

提示：通过执行 PDO 对象中的 query()方法返回的 PDOStatement 类对象，就是一个结果集对象；而通过执行 PDO 对象中的 prepare()方法返回的 PDOStatement 类对象，则是一个查询对象，本书约定用变量 $stmt 表示查询对象。

例 10.3 使用预处理语句插入记录的示例程序。

```
<?  require('conn.php');                //conn.php见 10.8.2节清单 10-19
    $sql="insert into lyb(title,content,author) values(?,?,?)";
                                        //用?号作占位符
    $stmt=$db->prepare($sql);           //准备执行查询
    $title='PDO预处理';    $content='这是插入的记录';    $author='西贝乐';
    $stmt->bindParam(1,$title);         //绑定第 1个参数
    $stmt->bindParam(2,$content);
    $stmt->bindParam(3,$author);
    $stmt->execute();                   //执行插入语句,将插入一条记录
echo '新插入记录的 ID是：'.$db->lastInsertId();
                   //如果要再插入记录,只要添加下面的代码即可
    $title='第二条';    $content='第二次插入的记录';    $author='书法家';
    $stmt->execute();                //再次执行重新绑定参数的准备语句,插入第二条记录
?>
```

例 10.4 使用预处理语句根据关键词查询的示例程序。

该程序将根据关键词查询结果,并将查询结果输出。代码如下:

```
<?  require('conn.php');          //conn.php 见 10.8.2 节清单 10-19
    $sql="select * from lyb where title like ?";  //用?号作占位符
    $stmt=$db->prepare($sql);                     //准备执行查询
    $title='进口';
    $stmt->execute(array("%$title%"));            //执行查询的同时绑定参数
    $row=$stmt->fetch(1);         //以关联数组的形式将结果集中第 1 条记录取出
    var_dump($row);                               //输出数组
    echo $row['title'];
?>
```

运行结果如下:

```
object(PDORow)#3 (9) { ["queryString"]=>string(36) "select * from lyb where title like ?" ["ID"]=>
string(3) "178" ["title"]=>string(11) "好进口红酒" ["content"]=>string(18) "返回梵蒂冈的航天员"
["author"]=>string(8) "回家看看" ["email"]=>string(9) "yaopi@163.com" ["ip"]=>string(9) "127.0.0.
1" }   好进口红酒
```

提示:在 SQL 语句中有 like 的情况下,占位符的正确写法是:"select * from lyb where title like ?",错误的写法是:"select * from lyb where title like '%?%'"。因为占位符必须用于整个值的位置,在绑定参数时再给关键词两边加"%"号。

10.9 用 PDO 制作博客网站

博客是一种个人展示网站,是继 Email、QQ、论坛之后出现的第四种网络交流方式。通过博客,用户可以方便地建立起个性化的私密空间,并将该空间的内容有选择性的与他人进行交流、沟通。博客一般包括主人的日志、相册、留言等模块。图 10-46 是一个简单博客网站的首页。

10.9.1 数据库的设计

根据博客网站的功能需求,我们将博客中的日志、相册、评论和留言分别保存在一张单独的表中。此外,为了让博客的主人可以登录后台对博客中的内容进行维护,还创建了一张管理员表,用来保存管理员的账号和密码信息。下面是博客网站数据库中包含的 5 个表的结构。

数据库名:Blog。

- 日志表:article(ID, title, content,author, date, hits, class)
- 相册表:album(id, title, author, desc, pic)
- 评论表:comment(id, articleid, author,content,date)
- 留言表:lyb(ID, title, content, author, email, ip, date, sex)
- 管理表:admin(id,user,password)



图 10-46　博客网站的首页

创建好数据库后，我们可以在数据库的每个表中添加几行记录作为测试数据。接下来，在 Dreamweaver 中新建一个动态站点 blog，再在站点目录下新建一个数据库连接文件 conn.php，代码如下（本例采用 PDO 方法连接数据库），这样就创建了网站与数据库的连接。

```
<?  $dsn="mysql:host=localhost;dbname=Blog";
    $db=new PDO($dsn,'root','111');           //连接数据库
    $db->query('set names gb2312');           //设置字符集
?>
```

10.9.2　首页的制作

制作首页的步骤是：首先制作一个静态页面，然后在该页面的各个栏目中绑定动态数据。

1. 制作静态页面

制作动态网站的第一步是设计静态网页，根据如图 10-46 所示的首页效果图，编写该首页的静态 HTML 代码，并保存为 index.php。结构代码如下（body 元素中的代码）：

```
<div id="top">                  <!--网页头部开始-->
  <div id="top_txt"><a href="javascript:addFav('博客网站示例');" title="添加到收藏夹">收藏
本页</a> |<a href="mailto:zh@qq.com" title="给站长发邮件">联系站长</a></div>
</div>
<div id="vi">
  <div id="tt">成功没有早晚<br />努力就有收获</div>
</div>           <!--网页头部结束-->
<div id="nav">                  <!--网页导航开始-->
  <ul>
      <li><a href="index.htm" target="_self">首页</a></li>
      <li class="bar"> |</li>
      <li><a href="article.htm" target="_self">日志</a></li>
  ……<li class="bar"> |</li>
      <li><a href="msg.htm" target="_self">留言</a></li>
  </ul>
</div>           <!--网页导航结束-->
<div id="main">              <!--网页主体开始-->
  <div id="left">            <!--左侧栏开始-->
    <h4> |最新留言</h4>
    <ul>
      <li>◆多拍点相片啊,大家分享一下。</li>
       ……
      <li>◆恭喜啊,个人网站终于开张了,下次记得请我们吃饭啊。</li>
    </ul>
  </div>           <!--左侧栏结束-->
  <div id="right">                <!--右侧栏开始-->
    <h4> |最新日志</h4>
      <span class="date">2011/01/25</span>
    <h5>Internet技术的应用</h5>
    <br /><p>在信息技术发达的今天,…</p>
    <hr />
        <span class="date">2011/01/23</span>
          <h5>网页技术学前班</h5>
    <br /><p>Internet,中文称为国际互联网。…</p>
  </div>                   <!--右侧栏结束-->
  <div id="photo">                <!--相册栏开始-->
    <h4> |最新相片</h4>
    <div id="photo_img">
      <img src="img/1_01.jpg" /><img src="img/1_02.jpg" />
      <img src="img/1_03.jpg" /><img src="img/1_04.jpg" />
      <img src="img/1_05.jpg" /><h5>森林美景</h5>
      <h5>地球景色</h5>  <h5>世界遗产</h5><h5>校园一角</h5>
      <h5>江边</h5>
    </div>  </div>           <!--相册栏结束-->
```

```
</div>            <!--网页主体结束结束-->
<div id="bt">本网站版权为 博主灰灰熊 所有<br /><span id="sysmsg"></span>
</div>
```

博客的首页需要将最新的日志、留言和相片显示出来，其原理主要是输出数据表中的记录。输出记录的步骤是：

(1) 创建结果集；

(2) 读取记录到数组中；

(3) 输出数组元素的值到页面上。

如果需要输出多条记录，可使用循环语句。

2. 最新留言的数据绑定

观察一下，最新留言需要输出 8 条留言表(lyb)中最新留言的标题(title)字段。为此，需要先创建结果集，将 lyb 表中的记录读取到结果集中来。可在网页的顶部插入如下代码：

```
<?  require('conn.php');
    $result=$db->query("select * from lyb order by ID desc");
?>
```

然后就可输出结果集中的字段到页面相应区域，这需要找到静态网页中需要被替换成动态字段的代码。找到如下代码：

```
<div id="left">
    <h4>|最新留言</h4>
    <ul>
        <li>◆多拍点相片啊，大家分享一下。</li>
                    …
        <li>◆恭喜啊，个人网站终于开张了，下次记得请我们吃饭啊。</li>
    </ul>
</div>
```

可发现，一条留言标题对应一个 li 元素。只要采用循环语句循环输出 8 个 li 元素，就能输出 8 条记录中的留言标题了。为此，先将 ul 中的 li 元素只保留一个，然后将这个 li 元素放到循环语句中循环输出 8 次，再将 li 元素的内容改为<?= $row["title"] ?>。修改后的代码如下：

```
<div id="left">
    <h4>|最新留言</h4>
    <ul>
    <?    if($result->rowCount()>0){
        for($i=0;$i<8;$i++){
            $row=$result->fetch(1);     ?>
                <li>◆<?=$row["title"] ?></li>
```

```
            <? }}
        else echo "<p>目前还没有用户留言</p>";
    ?>      </ul></div>
```

提示：上述代码中含有判断留言记录是否为空的语句。这样将使程序具有容错性，即使留言表中记录为空，程序也不会出错。

3. 最新日志的数据绑定

最新日志版块输出了 article 表中的两条最新日志。每条日志分别输出了标题、日期和内容，分别对应 title、content 和 date 字段。找到静态页面中的如下代码：

```
<h4>|最新日志</h4>
        <span class="date">2011/01/25</span>
        <h5>Internet技术的应用</h5>
    <br />
    <p>在信息技术发达的今天,…</p>
    <hr />
        <span class="date">2011/01/23</span>
          <h5>网页技术学前班</h5>
    <br />
    <p>Internet,中文称为国际互联网。…</p>
  </div>                    <!--右侧栏结束-->
```

可以将两条日志的代码只保留一条，使用循环的方式循环输出两条，然后在相应的位置分别嵌入要显示的动态字段。修改后的代码如下：

```
<h4>|最新日志</h4>
    <?$result=$db->query("select * from article order by ID desc limit 2");
     if($result->rowCount()>0){
while($row=$result->fetch(1)){          ?>
      <span class="date"><?=$row["date"] ?></span>
    <h5><?=$row["title"] ?></h5>
    <br />
    <p><a href="article.php?id=<?=$row["ID"] ?>"><?=$row["content"] ?></a></p>
    <hr />
    <?        }}         ?>
```

提示：上述代码使用“limit 2”子句使结果集中只包含两条记录。

4. 最新照片的数据绑定

在网页上如果要输出照片，一般是通过将照片的 URL 地址放入到 img 标记的 src 属性中来输出。由于每张照片的 URL 地址已保存到了数据表 album 的 pic 字段中，直接输出该 pic 字段值到 src 属性中即可。

```
<div id="photo">
```

```
<h4>|最新照片</h4>
<div id="photo_img">
  <img src="img/l_01.jpg" />
  <img src="img/l_02.jpg" />   <img src="img/l_03.jpg" />
  <img src="img/l_04.jpg" />   <img src="img/l_05.jpg" />
  <h5>森林美景</h5>
  <h5>地球景色</h5>        <h5>世界遗产</h5>
  <h5>校园一角</h5>        <h5>江边</h5>
</div>   </div>
```

将静态网页中5个重复的img元素只保留一个，使用循环语句循环输出5次，即得到了5张图片，图片标题对应的h5元素同样也只保留一个，再使用循环语句输出。代码如下：

```
  <div id="photo">
    <h4>|最新照片</h4>
    <div id="photo_img">
    <?$result=$db->query("select * from album order by ID desc limit 5");
     if($result->rowCount()>0){
while($row=$result->fetch(1)){          ?>
      <img src="img/<?=$row["pic"] ?>" />
<?     }}
    $result=$db->query("select * from album order by ID desc limit 5");
while($row=$result->fetch(1)){          ?>
      <h5><?=$row["title"] ?></h5>
      <?     }          ?>
    </div></div>
```

10.9.3 留言模块的制作

博客与访客互动主要依靠留言模块和评论模块。从功能上看，留言模块程序分为三部分，即显示留言（对应显示记录）、发表留言（对应添加记录）以及博客主人对留言的管理（对应修改和删除记录）。

留言模块的界面如图10-47所示。该界面上方显示留言列表，下方供访客发表留言。

1. 显示留言列表的实现

本例中将一条留言放置在一个div元素中，并用CSS设置样式。该div中可显示留言的标题、作者、作者图像、内容、发表时间和来自等信息，HTML代码如下：

```
<div id="liuyan"><img src="images/1.gif" style="float:left;"/>
    <h3>请教个问题</h3><p>作者：唐三彩</p>
    <p>内容：虚拟目录中…</p><p align="right">发表时间：
    2013-9-10 来自：127.0.0.1</p>   </div>
```

只要将这个div放在循环语句中循环输出，并将其中的静态文本替换成对应的动态



图 10-47　留言模块界面(msg.php)的效果

字段，就得到图 10-47 中留言列表界面，代码如下：

```
<? require('conn.php');
$result=$db->query("select * from lyb order by ID desc");
echo '共有'.$result->rowCount().'条留言';    ?>
<a href="Search.php">搜索留言</a>   <a href="login.htm">管理留言</a></p>
<?    if($result->rowCount()>0){
while($row=$result->fetch(1)){          ?>
        <div id="liuyan"><img src="images/<?=$row["sex"]?>.gif" style=
        "float:left;"/>
        <h3><?=$row["title"] ?></h3><p>作者:<?=$row["author"] ?></p>
        <p>内容:<?=$row["content"]?></p><p align="right">发表时间:
        <?=$row["date"]?>来自:<?=$row["ip"]?></p>   </div>
<?      }}
else echo "<p>目前还没有用户留言</p>";      ?>
```

该 div 元素调用的全部 CSS 代码如下：

```
<style type="text/css">
#liuyan {
    margin:8px auto;
    width:94%;
    border:1px solid #99CC99;
```

```
    padding:8px; }
#liuyan h3 {
    text-align:center;
    border-bottom:1px dashed gray;
    background:#FFFF99; }
#liuyan p {
    font:12px/1.6 "宋体";
    margin:2px; }
</style>
```

说明：表 lyb 的 sex 字段取值为 1 或 2。并在 images 目录下放置了两张图片 1.gif 和 2.gif。

2. 发表留言的实现

发表留言就是添加一条记录，其程序包括供访客发表留言的表单和接收表单数据的程序。表单代码如下：

```
<h2 align="center" style="background-color:#cc9; font-size:14px;">发表留言</h2>
<form method="post" action="submit.php">
  <table border="0" align="center" cellpadding="4" cellspacing="1"
  bgcolor="#666633">
  <tbody bgcolor="#ffffff">
    <tr><td width="125">留言主题:</td>
      <td width="475"><input type="text" name="title"></td></tr>
    <tr><td>留言人:</td>
      <td><input type="text" name="author">
           性别:男<input type="radio" name="sex" value="2" />
           女:<input type="radio" name="sex" value="1" /></td></tr>
    <tr><td>联系方式:</td>
      <td><input type="text" name="email"></td></tr>
    <tr><td>留言内容:</td>
      <td><textarea name="content" cols="30" rows="3"></textarea></td></tr>
        <tr><td></td>
      <td><input type="submit" name="Submit" value="提 交"></td></tr>
    </tbody></table></form>
```

接收表单数据并将数据作为一条记录添加 lyb 表的程序如下，本例采用 PDO 的预处理语句执行插入记录的 SQL 语句。插入完成后自动转到留言列表页面 msg.php。

```
<?  $title=$_POST["title"];          $author=$_POST["author"];
$sex=$_POST["sex"];      $content=$_POST["content"];
$email=$_POST["email"];      $date=date("Y-m-d H:i:s");
$ip=$_SERVER['REMOTE_ADDR'];
require('conn.php');
$sql="insert into lyb(title,author,content,sex,date,ip) values(?,?,?,?,?,?)";
```

```
$stmt=$db->prepare($sql);         //准备执行查询
$stmt->bindParam(1,$title);       //绑定参数
$stmt->bindParam(2,$author);          $stmt->bindParam(3,$content);
$stmt->bindParam(4,$sex);           $stmt->bindParam(5,$date);
$stmt->bindParam(6,$ip);
$stmt->execute();                 //执行插入命令
header("Location:msg.php");               ?>
```

10.9.4　博客后台登录的实现

1. 用户登录界面

在登录博客后台前，必须先验证用户的用户名和密码，以确定是否是真实的管理员，因此管理登录将链接到 login. htm，该页面代码如下，效果如图 10-48 所示。

```
<h4 align="center">博客后台用户登录</h4>
<form method="post" action="chklogin.php">
<table border="1"><tr><td align="center">用户名:</td>
    <td><input name="admin" type="text" size="12" /></td></tr>
    <tr><td>密  码 :</td>
        <td><input name="password" type="password" size="12" /></td></tr>
  <tr><td>验证码:  </td>
<td><input name="code" type="text" /><a href="javascript:reloadcode();">
<img src="code.php" id="code" border="0" align="absmiddle"/></a></td>
</tr>
<tr><td></td><td><input type="submit" value="登录" />
<a href="zhuce.htm">注册</a></td></tr>
</table></form>
<script>
function reloadcode(){  //刷新验证码的函数
    document.getElementById('code').src=
    'code.php?'+Math.random();
}    </script>
```

为了提高安全性，该用户登录模块设置了验证码验证功能，验证码的实现主要是：首先产业一个随机字符串，然后将该字符串保存到一个 Session 变量中。接着使用 PHP 内置的绘图函数将字符串绘制在一张图片中，最后通过 header 函数设置 PHP 文件以图片格式输出。code. php 的代码如下：



图 10-48　用户登录界面

```
<? session_start();
function random($len)  {
  $str="ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
  $s="";
```

```
    for($i=0;$i<$len;$i++){
            $s.=$str[rand(0,35)];     }
return strtoupper($s); }
$code=random(4);                       //随机生成 4 位长度的字符串
$_SESSION["code"]=$code;               //将验证码保存到 Session 变量中
$width=50;                             //验证码图片的宽度
$height=20;                            //验证码图片的高度
$im=imagecreate($width,$height);             //绘制验证码图片
$back=imagecolorallocate($im, 255, 255, 255);         //设置图片背景色为白色
$pix=imagecolorallocate($im,187,230,247);         //设置模糊点颜色
$font=imagecolorallocate($im,41,63,238);          //设置字符颜色
for($i=0;$i<1000;$i++) {                    //绘模糊作用的点
      imagesetpixel($im,rand(0,$width),rand(0,$height),$pix);
}
imagestring($im, 5, 7, 2,$code,$font);
imagerectangle($im,0,0,$width-1,$height-1,$font);
@header("Content-Type:image/png");      //设置 head 信息,输出图片
imagepng($im);                          //将图像流以 PNG 格式输出
imagedestroy($im);          ?>
```

2. 验证用户登录程序

验证用户登录程序是将用户输入的用户名和密码在 admin 表中进行查找,如果查找得到的结果集不为空,就表明有匹配的用户名和密码,验证通过,将转到后台管理页面 admin.php,并赋予用户一个 Session 变量,否则提示用户名或密码不正确。chklogin.php 的代码如下:

```
<? session_start();
require('conn.php');
$code2=$_POST['code'];
if($code2==$_SESSION["code"]){          //如果获取的验证码正确
    $admin=strip_tags(substr($_POST['admin'],0,32));
    $password=strip_tags(substr($_POST['password'],0,32));
    $crptpw=crypt(md5($password),md5($admin));
    $sql="select * from admin where user='$admin' and password='$crptpw'";
    $result=$db->query($sql);
    if($result->rowCount()==0){         //如果 admin 表中查不到对应的记录
        unset($_SESSION['admin']);
    echo "<SCRIPT>alert('输入的用户名或密码不正确!');history.go(-1)
</SCRIPT>";
    exit();}
else{
        $row=$result->fetch(1);
        $_SESSION['admin']=$row['user'];      //将用户名保存到 Session 变量中
        echo "<script>location.href='adminuser.php'</script>";    }    }
```

```
else echo "<script>alert('验证码不正确,请重新输入!');history.go(-1)
</script>";
?>
```

3. 验证用户是否已经登录的代码实现

在所有后台管理页面开头,都需要验证用户的 $_SESSION['admin']变量是否为空,如果为空,就表明没有登录,而是通过直接输入 admin. php 的网址进入的,此时不允许其访问后台,并将其引导至登录页面。

```
<?    session_start();
if($_SESSION['admin']==""){
    echo "<script>alert('您尚未登录或 Session 超时');location.href='login.htm'
</script>";
    exit();}          ?>
```

由此可见,Session 变量相当于系统给登录成功用户发的一张“票”,而所有后台管理页面都需要先验票才能决定是否允许用户访问,有了这张票才能访问后台管理页面。

10.9.5　博客用户注册模块的实现

博客用户注册模块的主要功能是:首先提供一个表单页面供用户输入要注册的用户名和密码等信息,如图 10-49 所示;用户提交表单后,检查用户输入的用户名是否已经被注册,如果未被注册,则允许注册;程序将对用户输入的密码进行加密,将加密后的密码连同用户名一起存放在数据表 admin 中。接收并处理用户注册信息的程序代码如下:



图 10-49　用户注册的表单界面

```
<? ------------------清单 10-4.php----------------------
require('conn.php');                    //连接数据库,使用 PDO 函数
    //获取表单信息,并过滤表单中的危险字符
$admin=mysql_real_escape_string(strip_tags(substr($_POST['admin'],0,32)));
$password=mysql_real_escape_string(strip_tags(substr($_POST['password'],0,32)));
$crptpw=crypt(md5($password),md5($admin));       //将用户提交的密码进行加密
$sql="select * from admin where user='$admin'";
$result=$db->query($sql);               //查询用户名是否已经被注册
if($result->rowCount()>0){
    echo "<script>alert('该用户名已经注册,请更换!');history.go(-1)</script>";
    exit();}
$sql="insert into admin(user,password) values('$admin','$crptpw')";
$affected=$db->exec($sql);              //将用户注册信息插入到 admin 表中
if($affected==1){
    echo "<script>alert('用户注册成功!');history.go(-1)</script>";
    exit();     }      ?>
```

10.9.6 用户管理模块的实现

用户管理模块包括用户管理主页面,如图 10-50 所示。在该页面上可以删除用户或修改用户密码。该程序首先执行查询显示 admin 表中所有的用户记录,每条用户记录的右边提供删除和修改密码链接,当单击“删除”链接时,转到 act.php 并根据记录 id 删除对应的用户记录。当单击“修改”链接时,在页面下方弹出修改密码的表单界面,要求用户输入原始密码和新密码,用户输入完成后,单击“确定”按钮将转到 act.php 页面。用户管理模块的代码如下:



图 10-50 用户管理主页面

```
<? ------------------------清单 10-6 adminuser.php--------------
require('conn.php');
```

```
$result=$db->query('select * from admin');      //执行查询创建结果集
    $result->setFetchMode(PDO::FETCH_ASSOC);
?><h3>后台用户管理</h3>
<table border="1" width="95%">
  <tr bgcolor="#e0e0e0">
    <th>ID</th><th>用户名</th><th>密码</th><th>删除</th><th>修改</th>
  </tr>
<? while($row=$result->fetch())    {          //读取一条记录到数组$row中
?><tr>
<td><?=$row['id'];?></td><td><?=$row['user'];?></td>
    <td><?=$row['password'];?></td>
        <td><a href="act.php?del=y&id=<?=$row['id'] ?>">删除</a></td>
        <td><a href="?mod=y&id=<?=$row['id'] ?>">修改</a></td>
        </tr>
<? } ?>
</table><p>共有<?=$result->rowCount()?>行</p>
<?
if($_GET['mod']=='y'){                          //如果是单击了"修改"链接
$id=intval($_GET['id']);                        //将获取的 id强制转换为整型
$sql="Select * from admin where ID=$id";        //查询待修改的记录
$result=$db->query($sql);                       //执行查询创建结果集
$result->setFetchMode(PDO::FETCH_ASSOC);
$row=$result->fetch();          //将待修改记录各字段的值存入数组中
?>
<h2 align="center">修改密码</h2>
<form method="post" action="act.php?mod=y&id=<?=$row['id'] ?>">
<table width="400" border="1" align="center" cellpadding="2">
<tr><td width="125">用户名:</td>
  <td width="275"><input type="text" name="user" value="<?=$row['user'] ?>"> * </td>
    </tr>
  <tr><td>原来密码:</td>
      <td><input type="text" name="oldpws"> * </td></tr>
    <tr><td>新密 码:</td>
      <td><input type="text" name="password"> * </td></tr>
    <tr><td>  </td><td><input type="submit" value="确 定"></td></tr>
  </table></form>
<? } ?>                     //if语句结束
```

10.9.7 删除用户与修改用户密码

删除用户的程序流程是：当在图 10-50 中单击“删除”链接时，就会转到 act.php 页面，并传递 del=y 和 id=n 两个 url 变量给 act.php，该程序根据 del=y 可判断是要执行删除操作，并根据 id 删除对应 id 的用户记录。

修改用户密码的程序流程是：当在图 10-50 中单击修改密码表单中的“确定”按钮

时，就会转到 act.php 页面，并传递 mod=y 和 id=n 两个 url 变量给 act.php，该程序根据 mod=y 可判断是要执行修改密码操作，修改密码操作又分为两步：首先查询用户输入的原来密码是否正确，如果不正确则退出修改程序；如果正确则将用户密码修改为新密码。act.php 的程序代码如下：

```
<? ------------------清单 10-7 act.php----------------------
require('conn.php');
$id=intval($_GET['id']);                    //将获取的 id强制转换为整型
if($_GET['del']=='y'){                      //如果是单击了“删除”链接
$sql="delete from admin Where ID=$id";      //根据 id删除记录
$affected=$db->exec($sql);
if($affected==1){                           //判断是否删除成功
    echo "<script>alert('该用户已经被删除!');location.href='adminuser.php';</script>";
    exit();
}}
if($_GET['mod']=='y'){                      //如果是修改密码操作
    //获取表单信息,并过滤表单中的危险字符
$admin=mysql_real_escape_string(strip_tags(substr($_POST['user'],0,32)));
$oldpws=mysql_real_escape_string(strip_tags(substr($_POST['oldpws'],0,32)));
$oldcrptpw=crypt(md5($oldpws),md5($admin));
$sql="select * from admin where user='$admin' and password='$oldcrptpw'";
$result=$db->query($sql);
if($result->rowCount()==0){                 //如果输入的用户名或原密码有误
    echo "<script>alert('您输入的原密码不正确!');history.go(-1)</script>";
    exit();
}
$password=mysql_real_escape_string(strip_tags(substr($_POST['password'],0,32)));
$crptpw=crypt(md5($password),md5($admin));
$sql="Update admin Set password='$crptpw' Where ID=$id";     //修改用户密码
$affected=$db->exec($sql);
if($affected==1){
    echo "<script>alert('用户密码修改成功!');location.href='adminuser.php';</script>";
    exit();
}}  ?>
```

习　题　10

一、练习题

1. PHP 哪个函数用于向 MySQL 数据库发送 SQL 语句？(　　)

   A. mysql_select_db　　　　B. mysql_connect

   C. mysql_query　　　　　　D. mysql_fetch_field

2. PHP 连接上 MySQL 之后，下列哪个函数配合循环可以得到指定表中的多条记录？（　　）

A. mysql_fetch_row　　B. mysql_select_db

C. mysql_query　　D. mysql_data_seek

3. “mysql_query("set names 'gb2312'");”，该行代码一般写在哪里最合适？（　　）

A. 创建结果集之前　　B. 创建结果集之后

C. 选择数据库之前　　D. 连接数据库服务器之前

4. 下列哪个函数可以将结果集的指针移动到指定的位置？（　　）

A. mysql_fetch_row　　B. mysql_fetch_assoc

C. mysql_query　　D. mysql_data_seek

5. PHP 连接 mysql 数据库的连接函数 mysql_connect 的第三个参数是（　　）。

A. 主机名　　B. 数据库密码　　C. 数据库用户名　　D. 报错信息

6. mysql_affected_rows()函数对下列哪个操作没有影响？（　　）

A. select　　B. delete　　C. update　　D. insert

7. mysql_insert_id()函数的作用是（　　）。

A. 返回下一次插入记录的 id 值　　B. 返回刚插入记录的自动增长的 id 值

C. 查看一共做过多少次 insert 操作　　D. 查看一共有多少条记录

8. mysqli 中返回结果集中记录总数的函数是（　　）。

A. fetch_row　　B. fetch_assoc　　C. num_rows　　D. field_count

9. 如果在 PHP 中使用 Oracle 数据库作为数据库服务器，应该在 PDO 中加载下面哪个驱动程序？（　　）

A. PDO_DBLIB　　B. PDO_MYSQL

C. PDO_OCI　　D. PDO_ORACLE

10. PDO 中要设置返回的结果集为关联数组形式，需使用（　　）。

A. fetch_row()　　B. fetch_assoc()　　C. fetch()　　D. fetch(2)

11. 如果在 PDO 中要执行已准备好的预处理语句，应使用下列哪个方法？（　　）

A. query()　　B. execute()　　C. exec()　　D. fetch()

12. 使用 mysql_query()函数发送 SELECT 语句时，执行成功将返回一个________，若执行查询失败则返回________。如果执行非 select 语句，执行成功时返回________，出错时返回________。

13. 在 mysqli 函数库中，从结果集中取出一行并返回一个关联数组的函数是________，返回一个索引数组的函数是________，返回一个混合数组的函数是________。

14. 在 mysqli 中，用来同时执行多个查询语句的函数是________。

15. 为了避免访问？MySQL？数据库时出现乱码现象，应在数据库连接文件中添加什么语句？

16. PHP 访问 MySQL 数据库，通常有哪三种方法？

17. MySQL 中使用 Select 语句查询时，要设置返回的行数可使用________子句。

18. 作为主键的 ID 字段应在“额外”中设置为________，新闻标题字段一般应设置为

________类型，新闻内容字段应设置为________类型。

19. ________语句可以向已存在的表中添加记录。

20. 在 MySQL 数据库中，varchar 和 char 两种数据类型有何区别？

21. 假设表 users 的结构如下，请写出“查询发帖数最多的 10 个人名字”的 SQL 语句。

users(id，username，posts，pass，email)

22. 修改 10.3.3 节中的 10-2.php，使它只显示 title 字段字段值，并且一行内显示 3 条记录的 title 字段。

23. 编写程序，将 lyb 表中的无重复的 title 字段值填充到一个下拉列表框中。